Giancarlo Baccolini   Carlo Offelli

MICROELABORATORI

NOTE DI HARDWARE

Giancarlo Baccolini
Carlo Offelli

MICROELABORATORI
Note di hardware

clup - milano

SOMMARIO

PRESENTAZIONE

Esiste una "seconda generazione" anche per i libri, oltre che per i calcolatori.
Questo è un libro della "seconda generazione": gli autori, infatti, si sono posti l'obiettivo di affrontare temi più avanzati, più immediatamente concretizzabili dal lettore, dopo la parte di inquadramento generale proposta con il precedente *Microelaboratori - Fondamenti.*

Va anzitutto sottolineato lo sforzo della CLUP nel proporre un volume all'interno di una serie attentamente progettata, che intende seguire lo studio e la crescita tecnica di quanti si occupano di microelettronica. E' indubbio che oggi sono reperibili sul mercato numerosi testi aventi come argomento i microprocessori e le problematiche associate: ma si tratta di libri singoli, mai di una serie completa, di complessità e rigore crescente.

In questo opera, non ci si ferma alla semplice elencazione dei problemi, del resto oggi abbastanza banale, ma si vuole dare al lettore uno strumento ampio e diversificato in grado di fornire delle risposte "reali" agli altrettanto reali problemi tecnici.
In quanti libri si parla di interfacciamento con memorie, in generale?
Ed in quanti libri ci si spinge ad illustrare come effettivamente si realizza una espansione di memoria, una organizzazione di un banco di memoria?
Questo è solo un accenno, appunto per sottolineare lo spirito da "seconda generazione" che caratterizza questo volume della serie Microelaboratori.

Infine, la leggibilità.
Posso affermare che non presenta particolari problemi addentrarsi in argomentazioni tecniche sofisticate mantenendosi a un livello di quasi completa leggibilità; il vero trasferimento tecnico-culturale presuppone un grande ed attento lavoro di chiarezza e concisione, col supporto di una serena conoscenza della materia. Il volume, infatti, vuole superare le barriere tra esperto e lettore in modo piano, non traumatico e aspro.
Pur non potendo annoverarmi tra gli autori, ho però avuto il piacere di partecipare alle prime fasi di definizione del testo, alla sua impostazione,

oserei dire, filosofica.
Per questo posso assumermi piena responsabilità di quanto detto in questa presentazione con la sicurezza che il successo di *Microelaboratori - Fondamenti* verrà senz'altro superato da questa opera, in attesa dei successivi volumi della serie, che conterranno per il lettore attento molte sorprese.

Aldo Cavalcoli

Milano, febbraio 1983

INTRODUZIONE

Questo lavoro vuole rappresentare la logica continuazione di quanto esposto nel precedente volume: "Microelaboratori - Fondamenti".
Il desiderio degli autori è di fornire quelle indicazioni di carattere generale atte alla realizzazione di un microelaboratore.
Sono perciò messe in risalto le modalità con cui si attuano gli scambi di informazione fra i vari dispositivi, funzionalmente distinti, presenti in un elaboratore, al fine di ricavarne quelle indicazioni che permettono di organizzare la struttura hardware richiesta in una qualsiasi applicazione.
Allo scopo sono analizzate le caratteristiche elettriche e funzionali di vari tipi di dispositivi di supporto al microprocessore, che stanno assumendo una importanza sempre crescente, in particolare quando presentano doti di programmabilità.
Dopo una analisi delle caratteristiche proprie dei canali attraverso i quali si effettuano gli scambi di informazione nell'interno dell'elaboratore, sono descritte le varie tecniche realizzative di banchi di memoria, sia RAM che ROM e le diverse modalità con cui si può effettuare la decodifica degli indirizzi. Successivamente sono descritte le varie possibilità di realizzazione di porte di ingresso e di uscita, sia nei casi più semplici che in quelli utilizzanti dispositivi programmabili.
Per quanto riguarda questi ultimi sono descritte porte programmabili di ricetrasmissione in parallelo, in serie, contatori e temporizzatori, interfacce per la gestione di tastiere e di visualizzatori.
Sono inoltre analizzati i problemi inerenti alle richieste di interruzione ed alla gestione dell'accesso diretto alla memoria da parte di dispositivi diversi dalla CPU.
Tutti i concetti esposti sono stati esemplificati con riferimento al microprocessore Z80, di ormai larghissima diffusione.

Capitolo 1
I BUS IN UN ELABORATORE A MICROPROCESSORE

1.1. Struttura di un sistema a microprocessore

Un sistema a microprocessore può essere schematizzato secondo quanto indicato nella fig. 1.1:

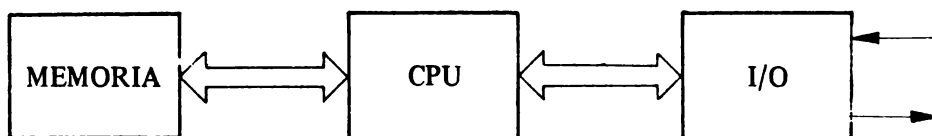



Fig. 1.1
Schema a blocchi di un elaboratore.

dove sono rappresentati i tre blocchi fondamentali che lo costituiscono e precisamente l'unità centrale, la memoria e l'interfaccia per il collegamento con gli organi di ingresso/uscita (I/O).
In genere la CPU è contenuta in un unico circuito integrato all'interno del quale si trovano tutti gli elementi che le consentono di svolgere le sue funzioni, come l'accumulatore, vari registri, l'unità logico-aritmetica, ecc.
Gli altri due blocchi sono invece composti da un certo numero di circuiti integrati a seconda del tipo e delle prestazioni dell'elaboratore che si vuole realizzare.
Nel blocco di memoria è compresa sia la memoria ROM, dove di solito sono memorizzati i programmi, che quella RAM, la quale è utilizzata per conservare i risultati parziali e finali delle elaborazioni eseguite dalla CPU. La capacità di tali memorie dipende ovviamente dai compiti richiesti all'elaboratore.

Per quanto riguarda il blocco di I/O la varietà dei dispositivi che lo possono costituire è molto ampia: si pensi ad esempio alle interfacce, notevolmente diverse fra loro, necessarie per il collegamento con le normali periferiche come la telescrivente (TTY), il floppy disk, il nastro magnetico, la stampante, lo schermo video, ecc., oltre ad eventuali interfacce particolari che sono necessarie quando si vuole utilizzare l'elaboratore per il controllo di un processo industriale.
Qualunque sia la struttura assunta dall'elaboratore che si vuole realizzare si deve risolvere il problema del trasferimento delle informazioni da un blocco all'altro di fig. 1.1.
Nella maggior parte dei casi il trasferimento avviene sempre fra l'unità centrale ed uno qualsiasi degli altri due blocchi, mentre non comunicano fra loro la memoria con il blocco di I/O, anche se, come si vedrà in seguito, si può organizzare in modo automatico questo tipo di operazione (DMA).

1.2. Organizzazione a bus

Il trasferimento di informazioni in un elaboratore può essere organizzato in modi diversi ma, in pratica, l'unico adottato per i notevoli vantaggi che presenta, è quello a bus.
Un bus può essere definito come un insieme di canali di collegamento, ognuno dei quali è composto da un certo numero di linee, cui accedono, in parallelo, i diversi dispositivi che nel loro insieme realizzano il microelaboratore.
In fig. 1.2 è indicata la struttura di un elaboratore in cui la comunicazione fra i diversi blocchi è attuata mediante un bus.

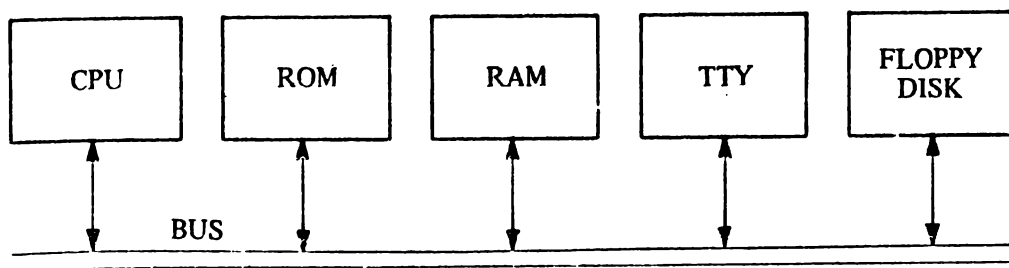



Fig. 1.2
Schema a blocchi di un elaboratore con architettura a BUS.

Il vantaggio di una tale struttura risiede nella facilità con cui si possono

aggiungere altri blocchi al sistema ove si abbia la necessità di aumentarne la potenzialità o variarne le caratteristiche.
I singoli blocchi, che sono collegati tra di loro tramite il bus, debbono presentare delle caratteristiche imposte da tale tipo di collegamento. E' indispensabile inoltre una particolare organizzazione per evitare che le informazioni scambiate tra due blocchi non siano alterate durante un trasferimento: tale situazione provocherebbe infatti l'introduzione di errori e quindi il malfunzionamento dell'intero sistema.
Poiché tutte le informazioni transitano attraverso uno stesso canale, comune a tutti i blocchi, che si trovano perciò collegati fra loro in parallelo, è necessario che in ogni istante sia attivo al massimo un solo trasmettitore, mentre possono essere attivi più ricevitori contemporaneamente, anche se in pratica quest'ultima situazione può verificarsi assai raramente.
Se avvenisse infatti la contemporanea attivazione di più trasmettitori, l'informazione presente sul canale risulterebbe indeterminata (bus contention) e, addirittura, ne potrebbero risultare dei guasti per i trasmettitori.
E' necessario quindi che in ogni istante uno solo dei blocchi costituenti il sistema assuma il controllo del bus e stabilisca inoltre il ruolo che deve essere svolto in ciascun intervallo di tempo dai rimanenti: il blocco che assume il controllo prende il nome di master, gli altri quello di slave.
In una struttura a bus possono anche essere presenti più blocchi in grado di funzionare da master, ma tale funzione può essere da questi assunta solo in tempi diversi.
Nello schema indicato in fig. 1.2 l'unico blocco in grado di assumere la funzione di master è la CPU, mentre tutti gli altri sono sempre slave. Ogni blocco è in grado di comunicare quindi solo con la CPU, che può assumere sia la funzione di trasmettitore che di ricevitore.
Se la CPU deve inviare una informazione ad uno degli slave presenti nel sistema, essa, innanzi tutto, notifica a tale slave, e solo a questo, che sarà interessato ad uno scambio di informazioni. Ciò può essere ottenuto tramite l'invio di una segnalazione che identifica univocamente lo slave interessato. E' inoltre necessario specificare che lo slave deve porsi nella funzione di ricevitore. Solo dopo che la CPU è certa che lo slave interessato si è posto in ricezione, potrà inviare l'informazione ad esso destinata.
Anche ogni scambio di informazioni fra due blocchi slave avviene per il tramite del master e cioè della CPU; se, ad esempio, si debbono inviare alla TTY dei dati contenuti nella memoria RAM, ciò può avvenire con due azioni distinte: un primo trasferimento di dati dalla RAM alla CPU ed un successivo trasferimento dalla CPU alla TTY.
Si vedranno in seguito le caratteristiche di blocchi particolari che permettono il trasferimento di dati anche senza interessare la CPU (DMA).

1.3. Suddivisione funzionale del bus

Tra le informazioni che i vari blocchi possono scambiarsi ci sono quelle, costituite da dati, che possono provenire dall'esterno, per essere elaborati direttamente dalla CPU o conservati in memoria RAM per successive elaborazioni; dati che possono anche consistere in risultati da inviare agli organi di uscita o da conservare nella memoria; dati che individuano le diverse istruzioni del programma che l'unità centrale deve eseguire. Tutte queste informazioni transitano attraverso un insieme di linee che prende il nome di bus dei dati (Data Bus).
Per far avvenire tali trasferimenti l'unità centrale deve precisare quale dei diversi blocchi di ingresso-uscita è interessato al trasferimento stesso, oppure quale cella di memoria contiene o dovrà contenere il dato da trasferire. Per effettuare questa selezione devono essere utilizzate altre informazioni, che nel loro insieme costituiscono l'indirizzo, mediante le quali è individuato, in modo univoco, l'altro dispositivo oltre alla CPU, interessato alla comunicazione. Tali informazioni sono presenti in un certo numero di linee che prende il nome di bus degli indirizzi (Address Bus).
Per quanto riguarda le modalità con cui può avvenire uno scambio di informazioni, esse dipendono sia dal blocco interessato, e quindi dalle sue caratteristiche funzionali, sia dal verso del trasferimento e dalla natura dei dati da trasferire. Tutte le segnalazioni utili a definire le modalità di trasferimento di informazioni fra i diversi blocchi prendono il nome di segnali di controllo, e transitano attraverso un certo numero di linee che costituiscono il bus di controllo (Control Bus).
In conclusione in un sistema a microprocessore si possono individuare diversi tipi di bus distinti per la loro funzione: il bus dei dati, il bus degli indirizzi e quello di controllo. Si tenga presente però che tale distinzione è di carattere funzionale e non fisico e permette solo di caratterizzare il tipo di informazioni che in ogni istante transitano attraverso uno di questi bus.
Per quanto riguarda la realizzazione di un bus si possono adottare varie soluzioni che dipendono e dal tipo di microprocessore utilizzato e dalle prestazioni che si richiedono al sistema.
In fig. 1.3 è riportata l'organizzazione del bus nei microprocessori Z80, Z8002, e 8080.
Nel caso di un microprocessore Z80 i tre bus sono fisicamente distinti fra loro.
Invece per il microprocessore Z8002 alcune linee assumono, in tempi diversi, la funzione di bus dei dati o degli indirizzi: il significato dei segnali presenti su tali linee è specificato da segnali del bus di controllo; è evidente la necessità di un hardware opportuno che provveda a separare ed utilizzare le informazioni, funzionalmente diverse, presenti sulle medesime li-

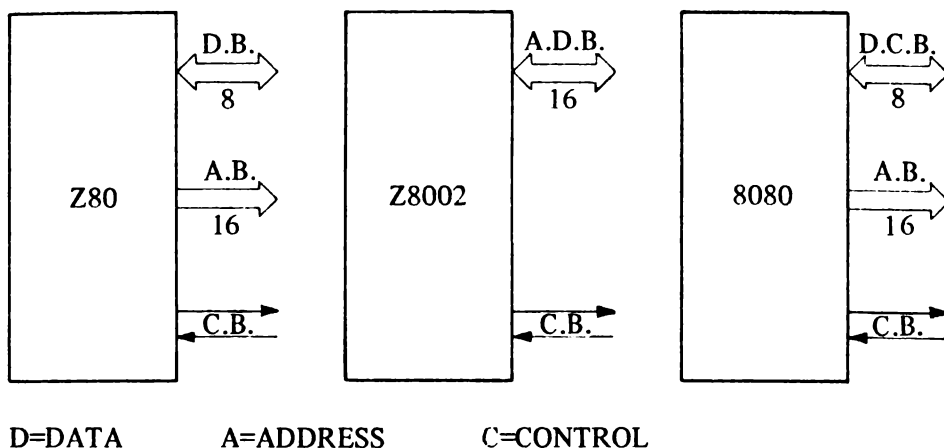



Fig. 1.3
Organizzazione dei bus nei microprocessori Z80, Z8002, 8080.

nee. In questo tipo di funzionamento si dice che il bus dei dati è "multiplexato" con il bus degli indirizzi.
Nel caso del microprocessore 8080 è invece il bus di controllo ad essere parzialmente multiplexato con quello dei dati in quanto le linee di quest'ultimo in certi istanti, contengono informazioni che funzionalmente appartengono al bus di controllo. Anche in questo caso altre linee del bus di controllo permettono di distinguere la funzione delle informazioni presenti, in tempi diversi, su stesse linee.
Per quanto riguarda il numero di linee che costituiscono i vari bus, esso dipende dal tipo di microprocessore utilizzato e dalle caratteristiche del sistema che si vuole realizzare.
In genere il bus dei dati è formato da tante linee quanti sono i bit presenti nell'accumulatore della CPU: 8 nello Z80, 8080, 8085, ecc.; 16 nello Z8001, Z8002, 8086, ecc.
Per quanto concerne invece il bus degli indirizzi esso è formato da un numero di linee che dipende dal massimo numero di locazioni indirizzabili dalla CPU: in genere con i microprocessori a 8 bit si hanno 16 linee di indirizzo con le quali si possono individuare fino a 64K (*) locazioni distinte.

(*) K = 1024.

Nel caso non sia necessaria tale capacità di indirizzamento il sistema che si vuole realizzare può presentare un numero di linee del bus degli indirizzi minore del massimo consentito dalla CPU.
Il bus di controllo è invece notevolmente diverso da caso a caso. Di solito quanto maggiore è il numero di linee presenti in tale bus tanto più semplice risulta l'organizzazione hardware dell'intero sistema. Ci si può rendere conto di ciò con un semplice esempio: per indicare che l'unità centrale vuole eseguire una operazione di lettura o di scrittura sulla memoria o su un organo di ingresso o di uscita, sono sufficienti due linee di controllo mediante le quali si fornisce in forma codificata, ad esempio secondo la tabella 1-1, l'informazione che individua una di queste quattro diverse operazioni.

Tabella 1-1

A	B	
0	0	lettura di memoria
0	1	lettura di un dispositivo di ingresso
1	0	scrittura di memoria
1	1	scrittura su un dispositivo di uscita

Per dedurre tali informazioni è necessario allora ricorrere ad un decodificatore con due ingressi e quattro uscite. Nel microprocessore Z80 sono presenti quattro distinti segnali di controllo, e precisamente MREQ/(*), IORQ/, RD/, e WR/, i quali individuano il tipo di operazione che sarà eseguita, senza la necessità di ricorrere al decodificatore.
Come visto per il bus degli indirizzi, al variare della realizzazione il bus di controllo non necessariamente presenta sempre lo stesso numero di linee, anche se si impiega lo stesso processore; infatti, sempre riferendosi allo Z80, è inutile prevedere la linea di controllo su cui transita il segnale per il rinfresco delle memorie dinamiche, RFSH/, se nel microelaboratore non sono impiegati tali tipi di memorie.
Il bus di controllo quindi può essere adattato a seconda del tipo di microelaboratore che si desidera realizzare.
In ogni caso il costruttore cerca di mettere a disposizione il maggior numero di segnali appartenenti a tale bus. E' questo il principale motivo per cui si ricorre all'uso di bus multiplexati. Infatti i contenitori dei microprocessori sono standardizzati e tipicamente sono presenti 40 piedini per i processori a 8 bit ed anche per alcuni a 16 bit. L'unico modo possibile per aumentare i segnali del bus di controllo senza aumentare il numero dei piedini del contenitore è quello del multiplexaggio dei vari bus, pur comportando tale tecnica l'uso di hardware aggiuntivo per la separazione delle diverse informazioni.

(*) La barra "/" dopo il nome di un segnale sta a significare che esso è attivo a livello logico basso.

In un microelaboratore realizzato secondo lo schema del tipo riportato in fig. 1.2 solo la CPU può assumere la funzione di master e quindi gli indirizzi per la selezione dei vari blocchi possono essere generati solo dal microprocessore. Per questo motivo il bus degli indirizzi presenta caratteristiche di unidirezionalità nel senso che le informazioni di indirizzo si propagano solo dal microprocessore verso il dispositivo cui sono destinate.
I dati da trasferire, invece, possono essere diretti dalla CPU verso altri blocchi o viceversa e quindi il bus dei dati è bidirezionale.
Nel bus di controllo transitano informazioni dirette dal master verso i vari slave e viceversa per cui anche questo bus deve presentare la caratteristica bidirezionalità. Essa però presenta degli aspetti diversi rispetto a quelli del bus dei dati: in genere infatti le varie linee che costituiscono il bus di controllo sono unidirezionali, ma alcune con direzione del master verso gli slave ed altre con direzione opposta; è per tale motivo che questo bus viene definito generalmente bidirezionale. Solo in pochi casi si hanno delle linee attraverso le quali possono transitare delle informazioni nelle due direzioni, cioè linee che devono presentare la caratteristica della bidirezionalità nel senso normalmente dato a tale termine.

1.4. Caratteristiche dei dispositivi collegati al bus

I vari dispositivi collegati al bus devono presentare delle ben precise caratteristiche in modo da ottenere, con un adatto controllo, un corretto funzionamento dell'intero sistema.
Per analizzare come si attua il collegamento al bus dei diversi blocchi costituenti il microelaboratore è sufficiente prendere in considerazione una qualsiasi sua linea ed individuare quali sono le caratteristiche elettriche che debbono presentare i singoli componenti collegati a tale linea, per permettere un corretto trasferimento di informazioni.
Molto spesso ognuno dei dispositivi collegati ad una linea generica del bus deve essere in grado sia di trasmettere che di ricevere informazioni: si pensi ad esempio ad una memoria RAM collegata ad una linea del bus dei dati. In altri casi invece un dispositivo deve essere in grado di svolgere una sola funzione che può essere di trasmettitore nel caso, ad esempio, delle memorie ROM collegate al bus dei dati, oppure di ricevitore, come per i componenti slave collegati al bus degli indirizzi. In genere quindi i dispositivi collegati al bus debbono presentare degli opportuni circuiti di interfaccia, inseriti fra i dispositivi stessi e le varie linee del bus, come è indicato in fig. 1.4.
In tali circuiti di interfaccia sono presenti a seconda dei casi i dispositivi per la ricezione o per la trasmissione delle informazioni o entrambi.
I ricevitori hanno accesso a tutte le informazioni che transitano nel bus cui sono collegati, ma le prendono in considerazione, cioè le fanno proseguire

all'interno del dispositivo, solamente quando sono destinate al dispositivo stesso. La principale caratteristica elettrica che devono presentare è quella di avere una impedenza di ingresso molto elevata, per non caricare il trasmettitore e quindi non influire in modo apprezzabile sui livelli logici dei segnali presenti sulla linea.

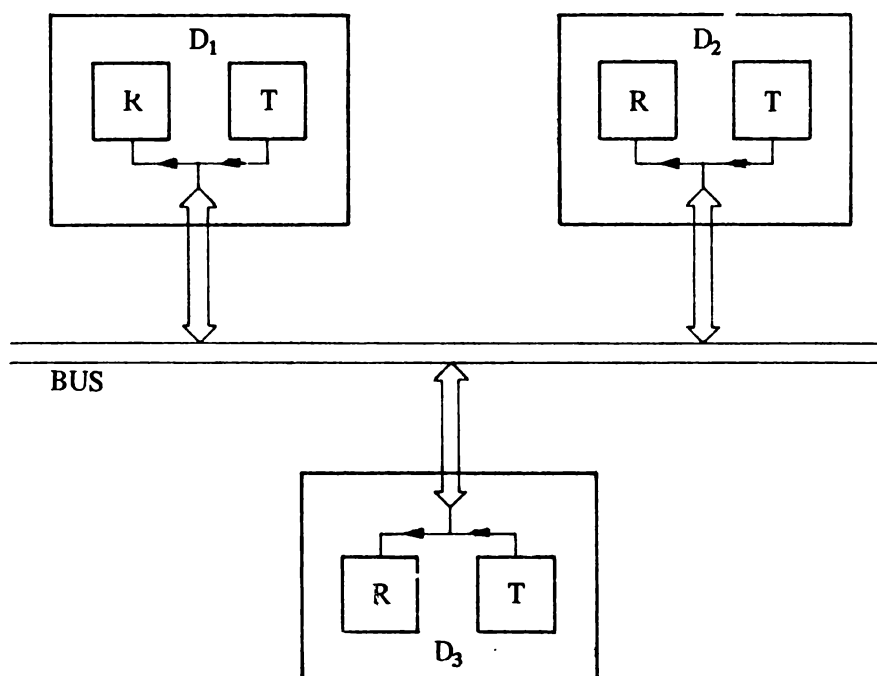



Fig. 1.4
Caratteristiche generali dei dispositivi collegati ad un bus.

Devono inoltre essere in grado di selezionare le informazioni ad essi indirizzate da quelle destinate ad altri ricevitori: questa funzione può essere ottenuta con varie tecniche, come sarà in seguito illustrato; in genere sono utilizzate opportune informazioni emesse dal master sui bus degli indirizzi e di controllo, eventualmente elaborate con dell'hardware aggiuntivo.
Come già detto, per un corretto funzionamento di una struttura bus è necessario che, in ogni istante, tra tutti i trasmettitori che sono connessi alla stessa linea ne sia attivo al massimo uno solo e che i restanti non influenzino in modo apprezzabile i valori di tensione imposti alla linea da questo unico trasmettitore.

Si possono distinguere quindi per ogni interfaccia di trasmissione due situazioni possibili: quella nella quale il trasmettitore invia delle informazioni sul bus, imponendo sulle linee i livelli logici di tensione alti e bassi, e quella, di inattività, nella quale esso non deve influenzare in alcun modo l'informazione presente sulle linee. Per ottenere quest'ultima caratteristica è indispensabile che l'interfaccia trasmettitrice presenti nello stato di inattività una impedenza di uscita molto elevata, in modo da poter essere considerata non connessa al bus e quindi permettere all'unico trasmettitore in funzione di imporre correttamente sulle linee i livelli di tensione alto o basso.
Allo scopo si possono adottare due diverse configurazioni per i circuiti di uscita dei trasmettitori: quella a collettore aperto (open collector) e quella a three-state.
Si analizzano ora le caratteristiche dei circuiti che adottano queste due diverse tecniche.

1.4.1. *Open collector*

Un circuito digitale si dice avere lo stadio di uscita a collettore aperto, quando il transistor di uscita funziona nella configurazione ad emettitore comune ed il suo collettore è collegato solo al piedino di uscita.
I vari collettori dei dispositivi trasmittenti che sono connessi ad una linea del bus sono collegati tra loro e, tramite un unico resistore, di pull-up, ad una unica tensione di polarizzazione. Nel caso estremamente semplice di più trasmettitori costituiti ognuno da un solo transistor, il collegamento ad una linea del bus è riportato nello schema di fig. 1.5.
Per un corretto funzionamento della struttura di fig. 1.5 devono essere soddisfatte determinate condizioni: nello stato di inattività ogni trasmettitore, nel caso in esame ogni transistor, deve presentare una impedenza di uscita molto alta. Per ottenere questo è necessario che i transistor di uscita siano interdetti e quindi che le tensioni imposte alle loro basi siano inferiori a quelle necessarie per renderli conduttori: conseguentemente la tensione sulla linea si porta a livello logico alto, pari a Vcc., se tutti i trasmettitori si trovano contemporaneamente nello stato di inattività. Quando entra in funzione uno, ed uno solo, trasmettitore, il livello assunto dalla linea del bus dipende solamente dal valore di tensione presente all'uscita di quel trasmettitore.
Nel caso di fig. 1.5 se una qualsiasi delle basi dei transistor si trova a livello logico alto, la linea del bus assume un livello basso in quanto quel transistor diventa conduttore.
Con le condizioni poste, le uniche configurazioni di ingresso possibili sono quelle riportate nella tab. 1-2 dove si può notare che sempre almeno due ingressi D_X si debbono trovare a livello basso perché la tensione sulla linea

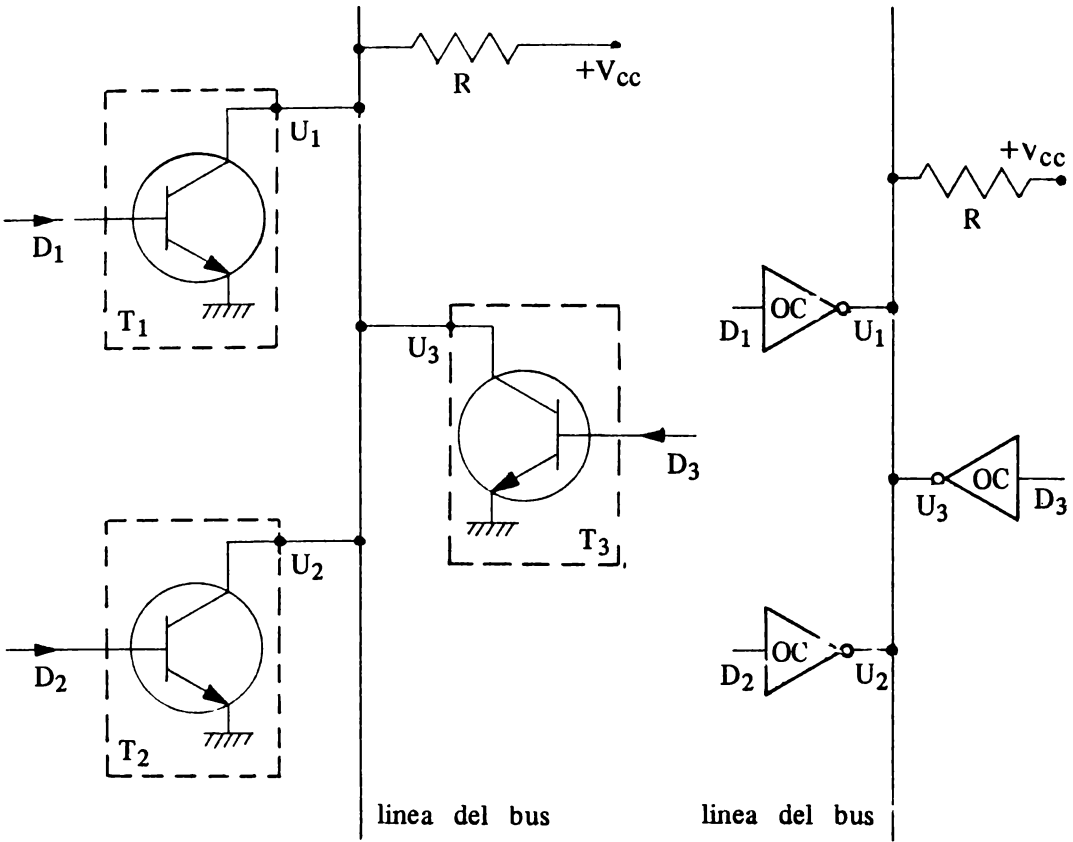



Fig. 1.5
Collegamento al bus di dispositivi con uscita "open collector".

Tabella 1-2

	D_1	D_2	D_3	LINEA
attivo T_3	0	0	0	1
	0	0	1	0
attivo T_2	0	0	0	1
	0	1	0	0
attivo T_1	0	0	0	1
	1	0	0	0

dipenda da quanto è presente sull'ingresso del trasmettitore attivo in quel-l'istante.
Con la tecnica dell'open collector il bus funziona in logica negata: i vari segnali di uscita sono nello stato inattivo quando si trovano a livello logico alto, pari alla tensione Vcc; la loro attivazione comporta che il bus assuma un livello basso a tensione uguale a quella di riferimento.
Per il corretto funzionamento di un bus con dispositivi open collector è necessario quindi che non solo sia attivo un solo trasmettitore alla volta, ma anche che gli ingressi dei dati di quelli non attivi si trovino in una ben precisa condizione. Invece di procedere ad un controllo direttamente tramite i dati di ingresso, si preferisce generalmente adottare degli schemi leggermente diversi, che consentono una più facile gestione. Una possibile struttura è riportata in fig. 1.6:

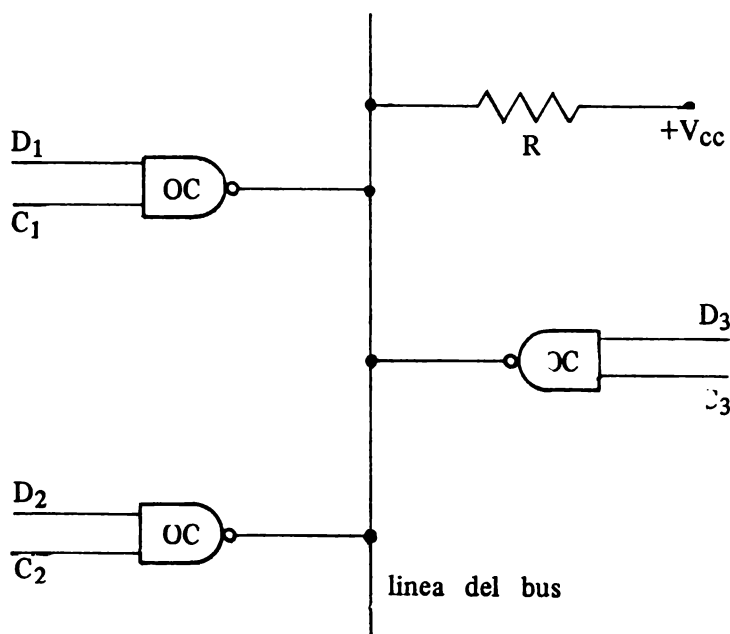



Fig. 1.6
Collegamento al bus di porte con uscita "open collector".

dove sono state utilizzate delle porte NAND, la cui uscita ovviamente è a open collector, per il collegamento al bus. In questo caso, oltre ai tre ingressi dei dati D_1, D_2, D_3, esistono anche i corrispondenti ingressi C1, C2, C3, che permettono di attuare, indipendentemente dal valore dei dati, il controllo necessario affinché si abbia un corretto funzionamento. Lo stato di inattività dei vari trasmettitori non è più stabilito dal valore assunto dal-

l'ingresso dati, come in fig. 1.5, ma dall'ingresso di controllo. In tabella 1-3 è riportata la tabella di funzionamento del circuito di fig. 1.6.

Tabella 1-3

		C_1	C_2	C_3	D_1	D_2	D_3	linea
inattivo		0	0	0	X	X	X	1
attivo T_1	{	1	0	0	1	X	X	0
		1	0	0	0	X	X	1
attivo T_2	{	0	1	0	X	1	X	0
		0	1	0	X	0	X	1
attivo T_3	{	0	0	1	X	X	1	0
		0	0	1	X	X	0	1

Come si può notare, con tutti i segnali di controllo a livello basso, la linea del bus si trova a livello logico alto indipendentemente dai dati presenti ai vari ingressi D_X. Se uno solo dei segnali di controllo è portato a livello alto, la linea del bus assume un valore logico che è funzione solo del livello presente all'ingresso dati del trasmettitore attivato.

1.4.2. *Three-state*

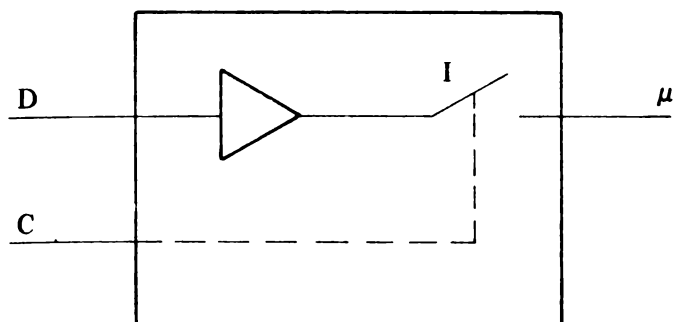
L'altra tecnica utilizzata nel funzionamento a bus è quella del three-state. Si dice che una porta presenta una uscita con caratteristica three-state quando la sua uscita non solo è in grado di assumere i due livelli logici alto e basso, in genere con una impedenza relativamente bassa, ma anche un terzo stato caratterizzato da un elevato valore di impedenza di uscita.
Da un punto di vista funzionale una porta con uscita three-state può essere schematizzata come in fig. 1.7.
Con l'interruttore I aperto l'uscita U si porta nello stato di alta impedenza ed il valore di tensione da essa assunto è indeterminato e comunque non dipendente da quanto presente sull'ingresso dati. L'interruttore I può essere chiuso inviando un valore logico opportuno all'ingresso di comando C, nel qual caso l'uscita diventa funzione dell'ingresso D (il valore logico assunto dall'uscita è uguale a quello di ingresso nell'esempio di fig. 1.7).
Il segnale di comando permette quindi di variare le caratteristiche di impedenza di uscita ed in particolare di imporre quando si deve passare nello stato di alta impedenza. Sono perciò soddisfatte le condizioni per poter utilizzare un tale tipo di porta in una configurazione a bus.

D	C	U
X	L	Hz
L	H	L
H	H	H

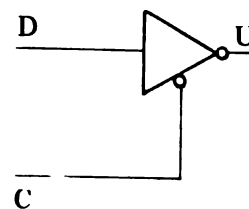
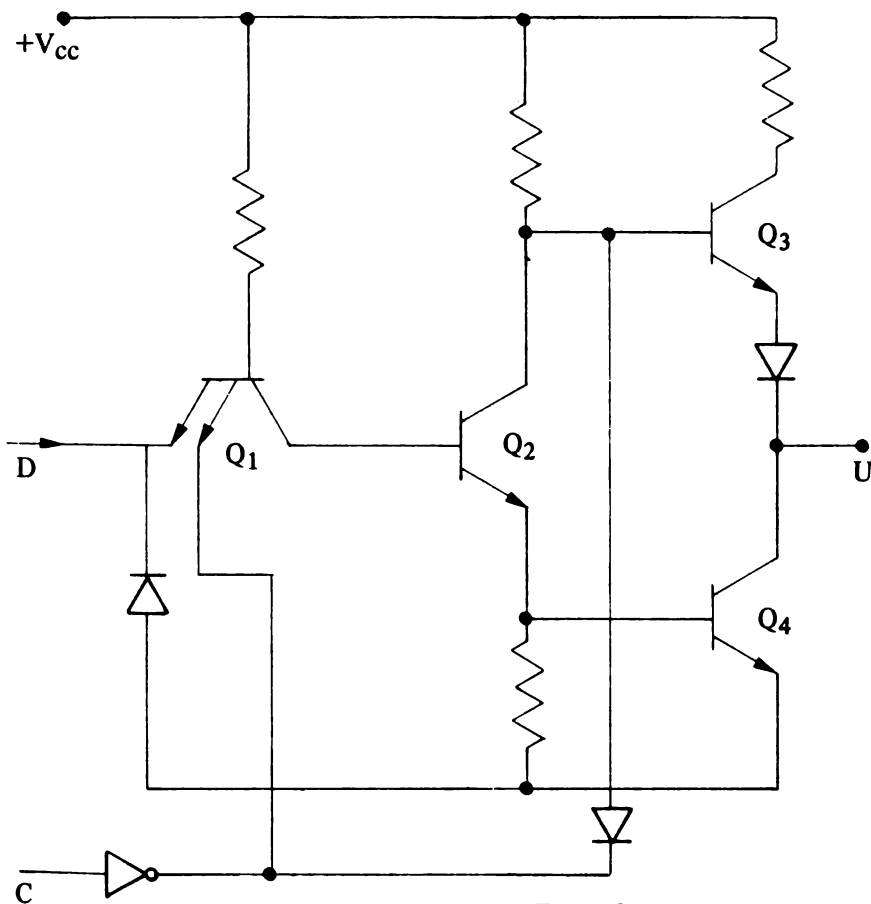
Fig. 1.7
Schema logico di una porta con uscita three-state e relativa tabella della verità.

D	C	U
X	H	Hz
H	L	L
L	L	H

Fig. 1.8
Schema di una porta NOT con uscita three-state, simbolo logico e tabella della verità.

Lo schema di una porta NOT three-state è riportato, assieme alla sua tabella di verità ed al simbolo normalmente utilizzato, in fig. 1.8.
Quando l'ingresso di comando C si trova a livello logico alto la base del transistor Q_3 assume un valore basso e quindi tale transistor risulta interdetto; il transistor Q_1 invece risulta conduttore e ciò comporta l'interdizione di Q_2 e quindi di Q_4. I due transistor di uscita si trovano entrambi interdetti per cui l'uscita presenta una impedenza molto elevata. Quando invece il segnale di comando è basso la porta funziona come un normale invertitore.
Esistono vari tipi di porte che presentano in uscita la caratteristica three-state; anche il valore logico che abilita o meno il trasmettitore può essere attivo a livello alto o basso.
Una possibile applicazione di tali porte three-state è riportata in fig. 1.9a dove si è, come al solito, indicato una sola linea del bus; la relativa tabella di funzionamento è riportata in tab. 1-4.

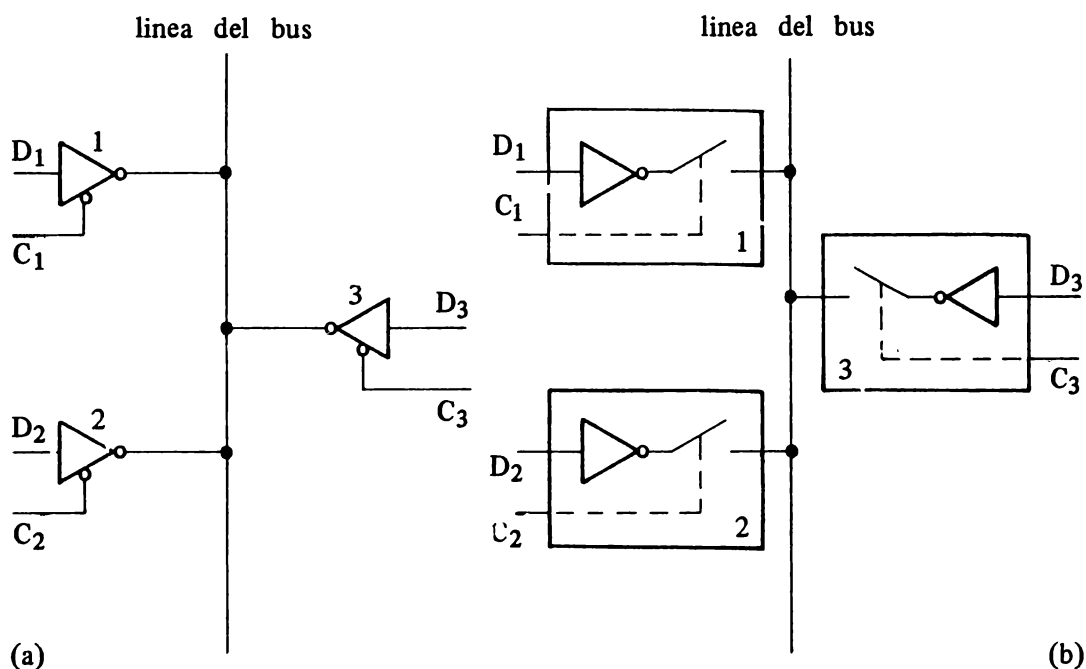



Fig. 1.9
Schema di collegamento (a) e schema logico (b) di porte con uscita "three-state" ad una linea del bus.

Tabella 1-4

	C_1	C_2	C_3	D_1	D_2	D_3	linea
	1	1	1	X	X	X	alta imped.
attivo 1	0	1	1	0	X	X	1
	0	1	1	1	X	X	0
attivo 2	1	0	1	X	0	X	1
	1	0	1	X	1	X	0
attivo 3	1	1	0	X	X	0	1
	1	1	0	X	X	1	0

Per un corretto funzionamento è necessaria una adatta gestione degli ingressi di comando; infatti solamente uno dei trasmettitori deve essere attivo, mentre gli altri devono essere lasciati nello stato di alta impedenza. In queste condizioni questi ultimi non influenzano l'unico trasmettitore in funzione in quanto possono logicamente essere considerati non connessi al bus come indicato in fig. 1.9b.
Si può presentare spesso il caso che un dispositivo da collegare al bus non presenti la sua uscita né in open collector né in three-state. Si interpone allora fra tale dispositivo ed il bus una porta che presenti una di tali caratteristiche, come è indicato in fig. 1.10.

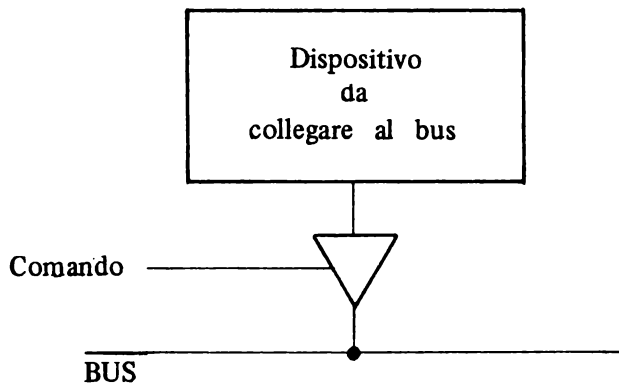



Fig. 1.10
Schema di collegamento di un generico dispositivo al bus mediante l'utilizzazione di una porta con uscita three-state.

Per quanto riguarda le caratteristiche dinamiche, in una porta three-state si hanno dei ritardi fra l'istante in cui si attiva o disattiva il comando e l'istante in cui effettivamente la porta diventa attiva o assume lo stato di alta impedenza; tali ritardi prendono il nome di tempo di abilitazione, t_a, e di disabilitazione, t_s, rispettivamente.
Nelle porte three-state è soddisfatta la relazione: $t_a > t_s$. In questo modo, quando si inviano i segnali di comando per disattivare un trasmettitore ed attivarne un altro, sicuramente questo diventa attivo dopo che il primo è stato disattivato e quindi si garantiscono le condizioni di corretto funzionamento del bus.
A volte si trovano delle strutture a bus con porte three-state dove è utilizzata una resistenza con uno schema simile a quello usato per il caso dell'open collector, ma con funzione completamente diversa. Nel caso tutti i trasmettitori siano inattivi, se non ci fosse tale resistenza, non sarebbe determinato il valore assunto in questa situazione dalla linea del bus; per togliere questa incertezza si utilizza allora una resistenza che polarizza la linea ad una tensione di riferimento.
Quando invece uno dei trasmettitori entra in funzione la tensione sulla linea è imposta dalla porta attivata.

1.4.3. *Confronto tra open collector e three-state*

Il principale vantaggio dell'utilizzazione della tecnica open-collector consiste nel fatto che il valore della tensione di polarizzazione delle linee può essere abbastanza elevato e quindi si possono ottenere delle alte insensibilità ai disturbi eventualmente indotti nell'ambito del bus, specie se esso presenta una discreta lunghezza, come in genere accade in un sistema di elaborazione.
Per contro, una volta dimensionato un bus per un certo tipo di applicazione, se si presenta la necessità di aumentare il numero di dispositivi ad esso collegati, in genere occorre diminuire il valore delle resistenze di polarizzazione delle linee ed usare dispositivi open-collector con maggiore capacità di assorbire corrente a livello logico basso.
Per quanto riguarda il valore di tali resistenze si noti che questo deve essere scelto con un giusto compromesso tra velocità di risposta e potenza dissipata nei dispositivi open-collector. Quanto appena detto riguarda le commutazioni della linea dal valore logico basso a quello alto: in questo caso infatti l'evoluzione della tensione sulla linea è, in prima approssimazione, di tipo esponenziale, con una costante di tempo cui contribuiscono essenzialmente il valore della resistenza di polarizzazione e quello delle capacità (proprie della linea, dei dispositivi ad essa collegati e quelle parassite) che essa deve caricare. Per diminuire tale costante di tempo, dato che le capacità sono imposte dalla configurazione fisica del bus e dai dispositivi presenti, l'unica grandezza su cui il progettista può agire è allora solo la

resistenza di polarizzazione; se il suo valore è piccolo, ogni dispositivo trasmettitore deve poter assorbire una corrente elevata e quindi deve poter dissipare, nello stato di conduzione del transistor di uscita, una maggior potenza.
Inoltre la notevole diversità dei valori di impedenza di uscita dei dispositivi open-collector, a seconda del valore logico generato, rende difficile evitare riflessioni dovute alle commutazioni; occorre allora aspettare che i valori di tensione si stabilizzino, e ciò comporta una ulteriore diminuzione della velocità apparente di commutazione del bus.
Gli inconvenienti accennati per il caso dell'open-collector sono in pratica inesistenti quando si utilizzino dispositivi di tipo three-state: per questi infatti l'impedenza di uscita, quando sono attivi, è praticamente costante e di valore ($\sim$ 100 Ω) prossimo a quello delle linee del bus. Sono perciò notevolmente attenuati i fenomeni di riflessione sulle linee ed è quindi più elevata la massima frequenza di commutazione possibile.
E' da sottolineare però che, nel caso di dispositivi three-state, se si verifica, per errore, che più di un trasmettitore piloti le linee del bus, il risultato è che le porte three-state hanno una buona probabilità di danneggiarsi (si pensi al caso in cui un trasmettitore vuole imporre un valore logico alto ed un altro un valore basso): ciò non accade nel caso di porte open-collector in quanto il trasmettitore del valore logico alto risulta essere fisicamente interdetto e quindi la massima corrente che può scorrere nel trasmettitore attivo a livello logico basso è proprio quella per cui esso era stato dimensionato.
Con la tecnica del three-state si può adottare, nell'ambito del bus, il tipo di logica, positiva o negativa, che più si ritiene conveniente, mentre nel caso dell'open-collector è utilizzata solo la logica negativa.

1.5. Caratteristiche hardware di un bus

La condizione fondamentale che deve essere soddisfatta da un bus è che le informazioni che partono dal trasmettitore arrivino al ricevitore senza subire la minima alterazione: il canale di comunicazione non deve essere nel modo più assoluto sorgente di errori.
Questa esigenza è imposta anche dal fatto che generalmente non si organizza la comunicazione fra trasmettitore e ricevitore con tecniche che permettono di individuare eventuali errori (codifica delle informazioni, bit ridondanti di controllo, ecc.) ma semplicemente il trasmettitore invia le informazioni ed il ricevitore acquisisce quanto arriva al suo ingresso. I criteri adottati per ottenere questa sicurezza nella trasmissione si fondano essenzialmente sulla conoscenza delle modalità di trasferimento di un segnale elettrico lungo una linea e sulle tecniche di schermatura al fine di evitare

che l'ambiente esterno possa influire sull'informazione trasmessa, introducendovi segnali spurii.
Una linea di trasmissione può essere schematizzata come indicato in forma semplificata nella fig. 1.11:

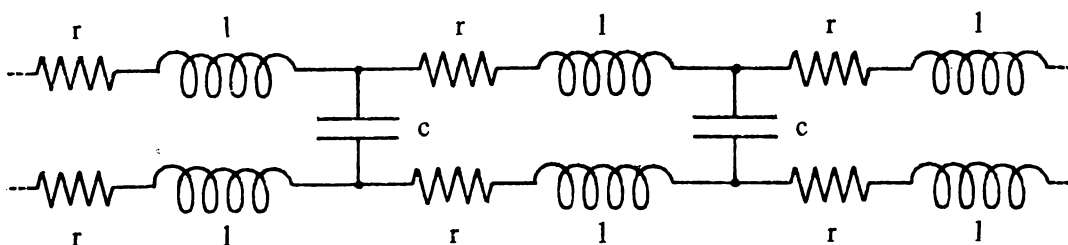



Fig. 1.11
Schematizzazione semplificata di una linea elettrica di trasmissione.

dove con r, l, e c si sono indicate rispettivamente la resistenza, l'induttanza e la capacità presentata dalla linea per unità di lunghezza. Una delle conseguenze della non idealità di una linea di trasmissione è una riduzione della velocità di propagazione delle grandezze elettriche che provoca un ritardo fra l'istante in cui l'informazione viene inviata dal trasmettitore e quello in cui essa è presente all'ingresso del ricevitore. Ciò comporta una velocità di trasmissione delle informazioni che non può superare determinati valori.
Inoltre i segnali di tensione e di corrente che si propagano lungo una linea di trasmissione, una volta arrivati alla sua estremità danno luogo ad un'onda riflessa, che si propaga in direzione opposta rispetto all'onda incidente; un successivo fenomeno di riflessione si presenta ancora quando l'onda riflessa arriva al generatore. L'ampiezza delle diverse onde di riflessione successive dipende dalle caratteristiche di impedenza presenti alle due estremità della linea.
Finché il fenomeno non si esaurisce, nei vari punti della linea si possono avere nello stesso istante dei valori diversi di tensione o di corrente: una eventuale lettura del livello logico presente nella linea durante questo transitorio può fornire un valore diverso da quello trasmesso, per cui si deve attendere la stabilizzazione dei vari segnali prima che un ricevitore sia in grado di leggere correttamente le informazioni ad esso inviate. Il transitorio dipende, oltre che dalle caratteristiche della linea, anche dal valore dell'impedenza che essa vede alle sue estremità. Nel caso tale impedenza sia uguale a quella caratteristica della linea (*) si annulla l'onda riflessa e quin-

(*) L'impedenza caratteristica di una linea è quella che essa presenta alle sue estremità nel caso sia di lunghezza infinita ed è uguale a $\sqrt{l/c}$.

di si riduce al valore minimo il transistorio con conseguente aumento della velocità di trasmissione delle informazioni. Per questo motivo ciascuna delle due estremità della linea è terminata mediante una resistenza collegata alla tensione di alimentazione, o a quella di riferimento, oppure, più frequentemente, ad una tensione generata mediante un partitore resistivo, come indicato in fig. 1.12.

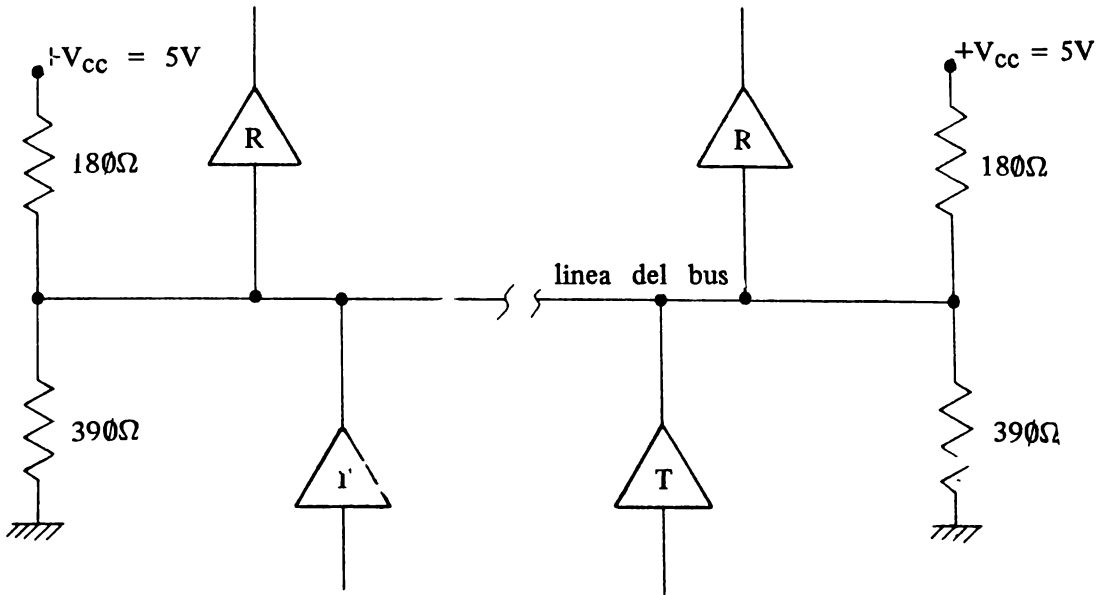



Fig. 1.12
Terminazioni di una linea del bus.

Per quanto riguarda il problema della prevenzione dei disturbi eventualmente indotti sulla linea da sorgenti di rumore, si adottano i criteri generalmente impiegati nei circuiti digitali come una efficiente schermatura, una opportuna dislocazione delle linee di trasmissione rispetto a sorgenti di rumore, l'uso di piani di massa per uno stabile riferimento delle tensioni, ecc.

1.6. Sincronizzazione dei segnali in un bus

Le informazioni che transitano sul bus rimangono costanti solamente per definiti intervalli di tempo e possono assumere significati diversi al variare dell'intervallo considerato ed al variare della sorgente e della destinazione.

Come si è già detto è compito del master coordinare il trasferimento delle informazioni stabilendone in modo univoco la sorgente, la destinazione ed il significato. In un microelaboratore il funzionamento del bus dipende dal microprocessore: poiché questo è sincronizzato da un opportuno segnale, detto appunto di sincronismo (clock), anche un microelaboratore presenta tale caratteristica. Ciò significa che tutte le diverse azioni che interessano un bus avvengono durante degli intervalli di tempo ben definiti, ognuno dei quali è di durata pari ad un certo numero intero di periodi di clock, numero che di solito varia col tipo di azione in corso.
Si voglia, come esempio, analizzare il ciclo di lettura di un byte da una memoria RAM; uno schema a blocchi parziale, dove sono riportati sia la CPU che la memoria con i bus di interconnessione ed i soli segnali che qui interessano, è rappresentato nella fig. 1.13.

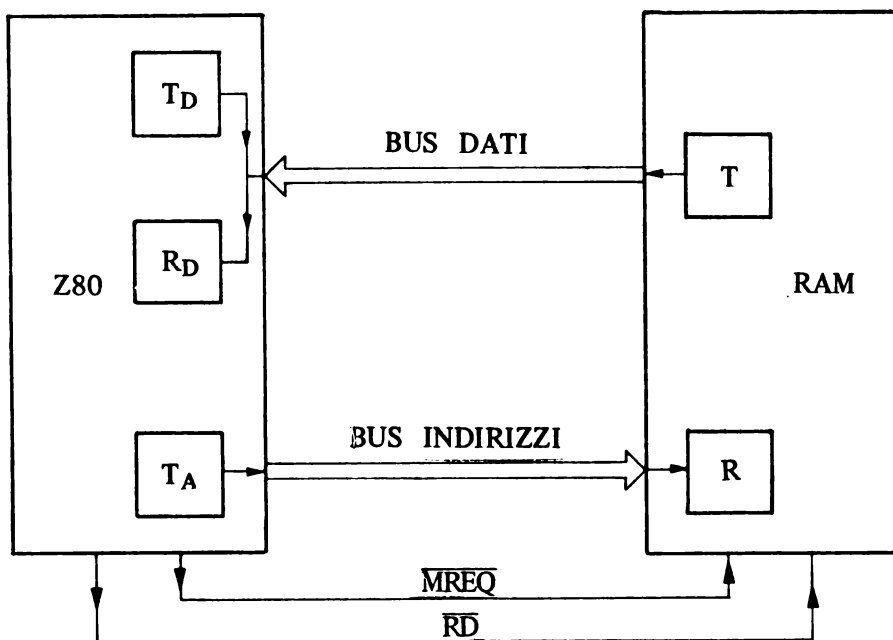



Fig. 1.13
Trasferimento delle informazioni per la lettura di un byte da una memoria.

Il microprocessore assume in questo caso la funzione di trasmettitore per quanto riguarda le informazioni relative al bus degli indirizzi; nei confronti del bus dei dati esso assume il ruolo di ricevitore, mentre il trasmettitore è la memoria RAM. La validità ed il significato delle informazioni presenti

sui bus sono forniti mediante i segnali del bus di controllo, anch'essi generati dal microprocessore.
Facendo riferimento allo Z80, il diagramma temporale di un ciclo di lettura da memoria è indicato nella fig. 1.14.

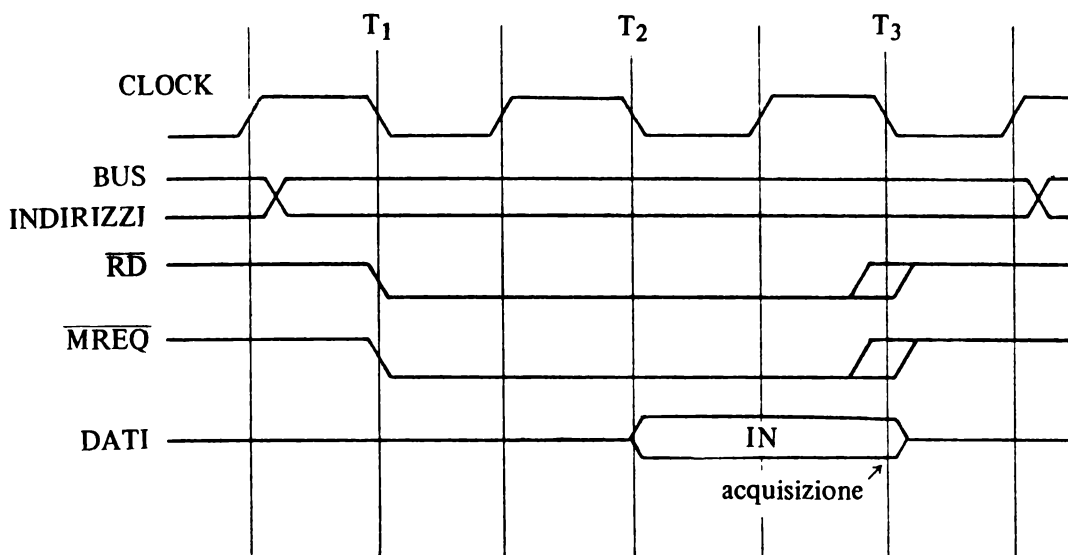



Fig. 1.14
Diagramma temporale di un ciclo di lettura di un byte dalla memoria.

Dopo un certo tempo dal fronte di salita del primo periodo di clock T_1, il microprocessore invia sul bus degli indirizzi l'informazione che permette di individuare in modo univoco la cella di memoria da dove prelevare il dato in essa contenuto. Anche se indicato solo in forma qualitativa nel diagramma temporale, è necessario un certo intervallo di tempo affinché i segnali sul bus degli indirizzi assumano dei valori stabili e quindi sia possibile individuare la cella di memoria desiderata. Si noti che fino al fronte di discesa di T_1 non sono utilizzate le informazioni presenti nel bus degli indirizzi in quanto finora nessuno dei componenti collegati al bus è stato avvertito, tramite un qualche segnale del bus di controllo, di entrare in attività.
Allo scopo esistono due segnali che permettono di gestire correttamente la lettura della memoria: il segnale MREQ/ (che è attivo a livello logico basso) ha la funzione di informare i dispositivi del sistema che sul bus degli

indirizzi ne è presente uno già sicuramente stabilizzato, e quindi utilizzabile, e tale indirizzo è riferito ad una cella di memoria; non è indicata, con tale segnalazione, quale operazione si intende compiere sulla memoria stessa.
Quest'ulteriore informazione è fornita mediante il segnale RD/ la cui attivazione precisa che si intende eseguire una lettura della cella di memoria individuata dall'indirizzo presente in quel momento sul bus omonimo.
A questo punto tutte le informazioni necessarie per l'operazione desiderata sono state fornite dal master, che finora ha assunto la funzione di trasmettitore mentre lo slave (la memoria) quello di ricevitore. Per il tipo di operazione desiderata la memoria deve assumere ora il ruolo di trasmettitore, mentre il microprocessore, che conserva ancora la funzione di master, diventa il ricevitore.
Non appena sono presenti i segnali di controllo, la memoria entra in attività e dopo un certo intervallo di tempo, dipendente dalle caratteristiche della memoria stessa, mette a disposizione sul bus dei dati quanto è contenuto nella cella indirizzata. Nella fig. 1.14 si è indicato, per semplicità, che il tempo necessario alla memoria per fornire il dato richiesto sia circa uguale ad un periodo di clock; si noti inoltre che non è fornita alcuna indicazione per informare gli eventuali altri componenti del sistema che l'attuale trasmettitore ha inviato una informazione sul bus, in quanto essi non sono interessati alle azioni che seguiranno.
Il microprocessore non ha ancora assunto il ruolo di ricevitore, per cui quanto presente sul data bus non viene da questi acquisito: durante il secondo periodo di clock, T_2, il microprocessore rimane infatti inattivo.
Il ruolo di ricevitore è assunto dal microprocessore in corrispondenza del fronte di discesa di T_3: in tale istante esso acquisisce quanto è presente sul bus dei dati; è compito della memoria far si che in tale istante il contenuto del bus corrisponda a quello della cella indirizzata.
Il master provvede quindi alla disattivazione dei segnali di controllo MREQ/ e RD/ e dal successivo periodo di clock procede alla esecuzione di un'altra azione.
Da quanto illustrato risulta che il funzionamento del bus è sincronizzato con il funzionamento del master, e solo in tal modo si ha un corretto trasferimento delle informazioni.
Si vuol far presente che lo schema di fig. 1.13 ha solo la funzione di illustrare le caratteristiche di un ciclo di lettura di una memoria e non il collegamento della stessa con l'unità centrale. Inoltre nella descrizione di tale ciclo non si è indicato come si deve organizzare il sistema quando il tempo di accesso della memoria supera certi valori.

1.7. Collegamento di dispositivi al bus

Uno dei principali vantaggi nell'utilizzare una struttura a bus è che con una certa facilità possono essere aggiunti al sistema nuovi dispositivi qualora si renda necessario aumentare o variare le prestazioni dell'elaboratore. Se non sono soddisfatte certe condizioni può però verificarsi che il collegamento di un nuovo dispositivo introduca delle alterazioni alle caratteristiche della struttura a bus e provochi un funzionamento errato dell'intero sistema.

Si può considerare come esempio una linea del bus alla quale sono collegati un certo numero di trasmettitori con uscita three-state, e di ricevitori, i quali presentino le caratteristiche della famiglia di circuiti integrati TTL (fig. 1.15).

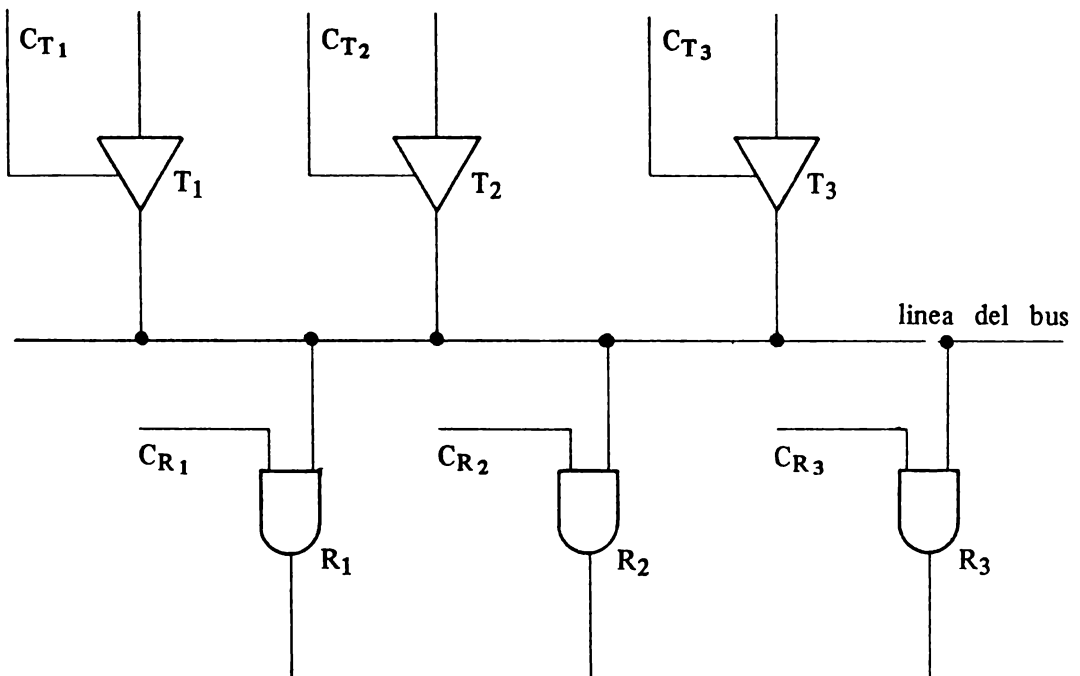



Fig. 1.15
Collegamento di più trasmettitori e ricevitori ad una linea del bus.

Si analizza ora il comportamento del circuito di fig. 1.15 prima in regime statico e poi in quello dinamico.

1.7.1. *Caratteristiche statiche*

Come è noto una tensione è considerata a livello logico alto (o basso) se si trova entro un determinato intervallo di valori.

Con riferimento all'ingresso ed all'uscita di una porta, si possono allora considerare quattro valori di tensione, come è indicato in forma schematica nella fig. 1.16.

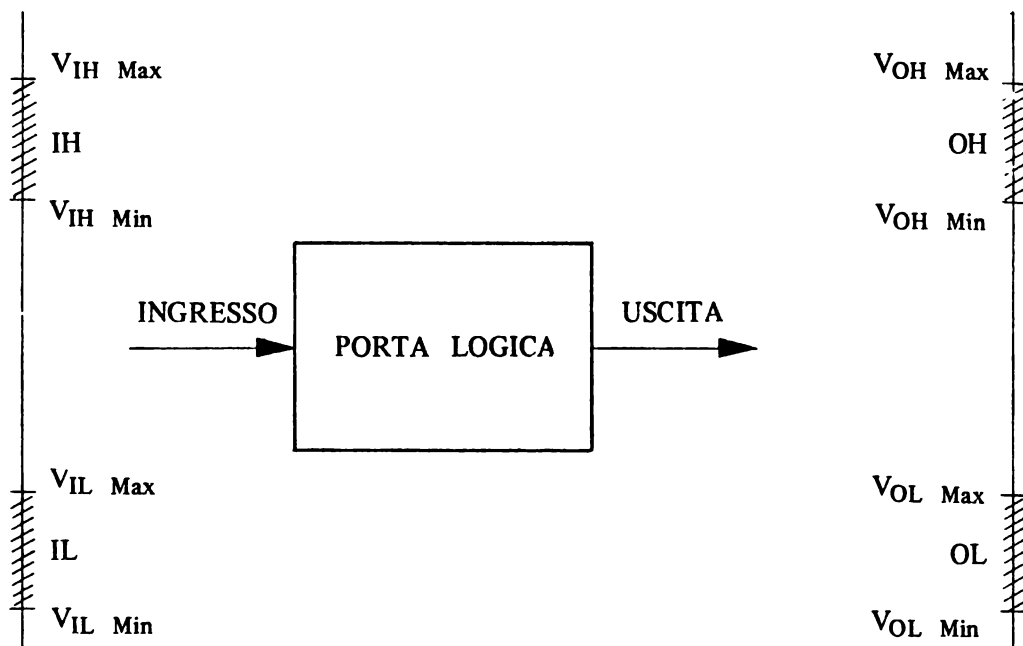



Fig. 1.16
Intervalli di validità dei livelli di tensione di ingresso e di uscita in una porta logica.

I valori di tensione, sia all'ingresso che all'uscita, devono mantenersi entro gli intervalli tratteggiati di fig. 1.16 perché siano considerati validi.
A seconda del livello di tensione assunto all'ingresso e all'uscita si ha sia in ingresso che in uscita una circolazione di corrente. E' necessario che essa si mantenga entro i valori stabiliti dal costruttore e che la tensione permanga entro le fascie di valori di fig. 1.16 perché l'informazione venga interpretata in modo corretto. Convenzionalmente si assume come positiva una corrente entrante nel dispositivo, come negativa se uscente. Si possono allora avere quattro diversi valori di corrente, come indicato in fig. 1.17, ognuno dei quali è compreso entro un certo intervallo di valori limitati da un valore minimo e massimo.
Nel caso delle uscite (TTL) di tipo three-state si deve considerare anche la corrente circolante quando l'uscita si trova nello stato di alta impedenza. Si può ritenere che tale corrente abbia la stessa intensità sia che l'uscita si trovi, a causa di un collegamento ad altri dispositivi, al valore alto oppure basso; essa però ha versi opposti ed in particolare con tensione a livello

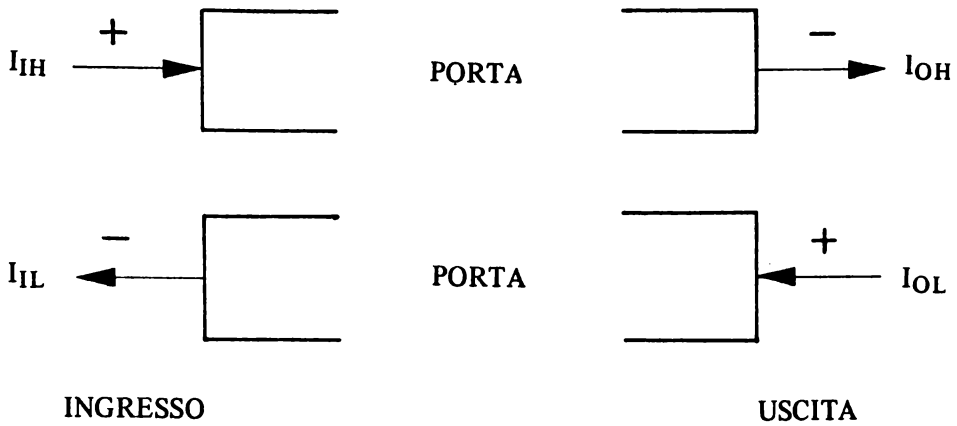



Fig. 1.17
Correnti di ingresso e di uscita in una porta logica.

alto si ha una corrente entrante, mentre è uscente se si è a livello basso, come indicato nella fig. 1.18, dove il simbolo I_{OLHZ} sta a significare corrente di uscita (O) con livello di tensione basso (L) nella condizione di alta impedenza (HZ) (in modo analogo per l'altro simbolo).

Fig. 1.18
Correnti di uscita in una porta three-state nello stato di alta impedenza (HZ).

In fig. 1.19 è riportato lo schema di fig. 1.15 con l'indicazione del verso delle correnti, nelle due situazioni di linea a livello alto e basso, nell'ipotesi che solamente il trasmettitore T_1 sia attivo, mentre gli altri si trovano nello stato di alta impedenza.

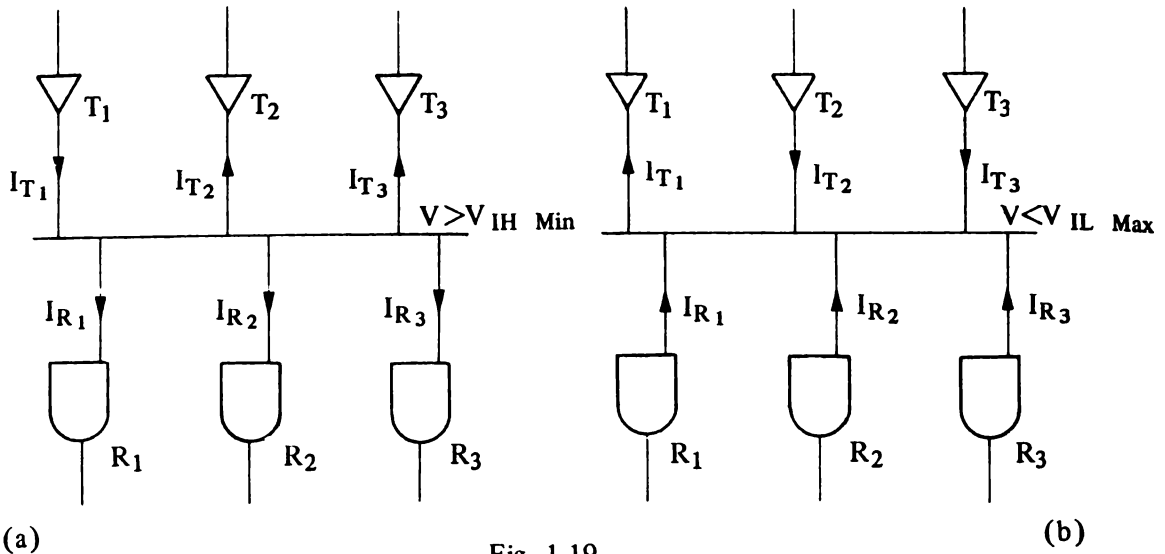



Fig. 1.19
Versi delle correnti con T_1 attivo e $V > V_{IHmin}$ (a) oppure $V < V_{ILmax}$ (b).

Affinché la linea assuma un valore di tensione tale da essere interpretato come valore logico alto è necessario che il trasmettitore in funzione sia in grado, oltre che di imporre tale valore, anche di erogare la corrente che viene assorbita dagli altri componenti collegati al bus e cioè:

$$| I_{T_1\ OH} | > I_{T_2} + I_{T_3} + .. + I_{IH\ R_1} + I_{IH\ R_2} + I_{IH\ R_3} + ...$$

In modo analogo affinché la linea assuma un valore di tensione che rientri nel campo del livello logico basso è necessario che il trasmettitore attivo sia in grado di assorbire la corrente fornita dagli altri dispositivi, e cioè deve essere soddisfatta la relazione

$$I_{OLT_1} > | I_{T_2} + I_{T_3} + ... + I_{ILR_1} + I_{ILR_2} + ... |$$

Se anche una sola di queste due condizioni non è soddisfatta si ha in linea un livello di tensione che non è compreso entro gli intervalli di valori fissati e quindi si verificano degli errori.
Come esempio si supponga che nello schema di fig. 1.15 il trasmettitore T_1 abbia le caratteristiche di uscita tipiche del microprocessore Z80.
Per esso la corrente di dispersione nella condizione three-state è di 40 μA e le correnti di uscita sono:

$$I_{OL_{min}} = + 1{,}9 \text{ mA}; \qquad I_{OH_{max}} = - 250\ \mu\text{A}$$

ai valori limite rispettivamente di $V_{OL\ max}$ e $V_{OH\ min}$, mentre per l'ingres-

so dei ricevitori si possono assumere dei valori tipici di:

$I_{IH} = + 40\ \mu A$; $I_{IL} = - 1{,}6\ mA$.

Dai valori forniti si può notare che il microprocessore Z80 è in grado di comandare sei porte TTL al valore logico alto ed una sola porta a quello basso, per cui lo schema di fig. 1.15 sicuramente provoca un funzionamento errato. Si noti che i valori indicati per il microprocessore Z80 sono abbastanza tipici e simili quindi a quelli presentati da altri tipi di microprocessore.
In conclusione dall'indagine fatta si ricava che è possibile collegare ad una linea di uscita dello Z80 un solo ricevitore di tipo TTL. Per ovviare a tale limitazione si deve interporre fra il processore e la linea del bus un dispositivo in grado di fornire la corrente necessaria a pilotare i vari ricevitori. Tale dispositivo prende il nome di buffer (o driver) ed è interposto fra ogni trasmettitore e la linea del bus, ove se ne presenti la necessità.
Un esempio di circuito integrato che potrebbe essere impiegato è il 74S244 per il quale ognuna delle 8 porte buffer che lo compongono presenta come caratteristiche di ingresso e di uscita i valori della fig. 1.20.

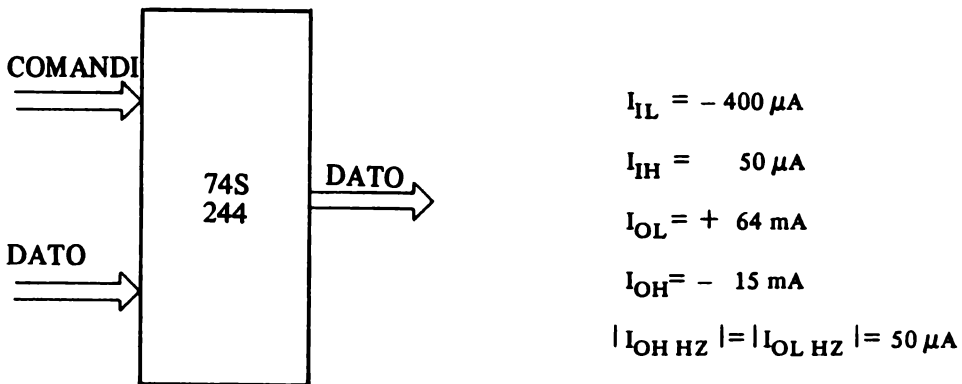



Fig. 1.20
Buffer 74S244: il valore delle correnti nelle diverse situazioni.

Dalle caratteristiche di ingresso di tale dispositivo si vede che sicuramente lo Z80 è in grado di pilotare questo buffer, e si può notare inoltre che la sua corrente di uscita è notevolmente superiore a quella fornibile dallo Z80 e quindi aumenta il numero di porte pilotabili quando tale buffer funziona da trasmettitore.
Il driver 74S244 ha un funzionamento unidirezionale; esistono dei driver integrati che permettono il trasferimento delle informazioni in due dire-

zioni e prendono il nome di buffer bidirezionali. Lo schema relativo alla funzione svolta per ogni linea del bus è indicato in fig. 1.21:

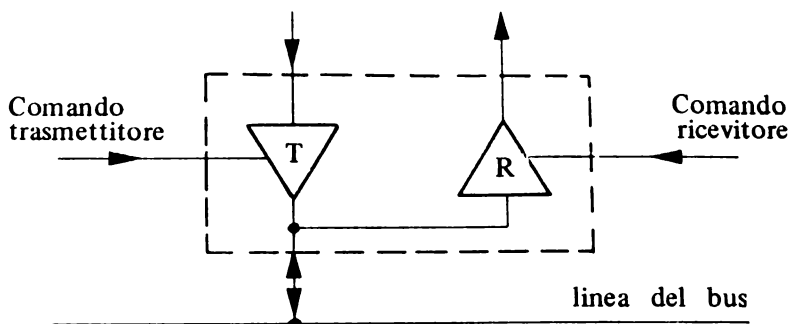



Fig. 1.21
Schema funzionale di un buffer bidirezionale.

Si ha una amplificazione di potenza sia in trasmissione che in ricezione: non sempre tale amplificazione ha lo stesso valore nelle due direzioni; opportuni comandi permettono di stabilire il verso di propagazione dei segnali sulle linee.
Per quanto riguarda invece la realizzazione di un bus con la tecnica dell'open-collector occorre determinare e quali debbano essere le caratteristiche di pilotaggio delle linee da parte di un generico trasmettitore e quanto debba valere la resistenza di polarizzazione; ciò al variare del numero di ricevitori e di trasmettitori presenti sulla stessa linea, per assicurare i valori di tensione che corrispondono a quelli fissati per i valori logici alto e basso.

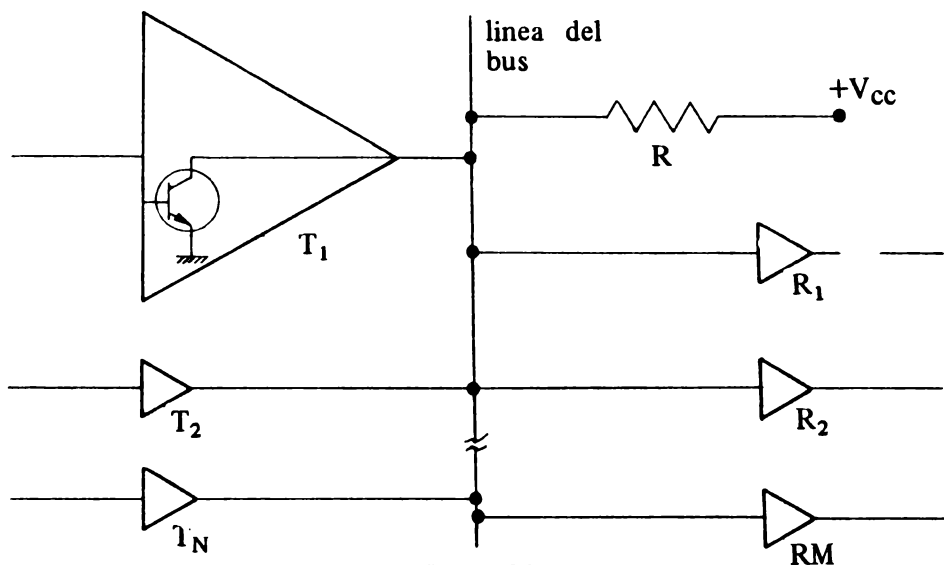



Fig. 1.22
Ricevitori e trasmettitori con uscita open collector collegati ad una linea del bus.

In fig. 1.22 è indicata schematicamente una linea cui sono collegati N trasmettitori ed M ricevitori; si supponga inoltre che si tratti di dispositivi appartenenti alla famiglia di circuiti integrati TTL. Il valore della resistenza R si può calcolare separatamente nei due casi in cui in linea è presente il livello logico basso o quello alto: la resistenza R avrà un valore compreso nell'intervallo di valori i cui estremi sono quelli che risultano dai calcoli eseguiti nei due casi distinti.
Per quanto riguarda la condizione che la linea assuma il valore logico alto, cioè nel caso che tutti i trasmettitori abbiano il transistor di uscita interdetto, siano I_{OH} la corrente di dispersione, attraverso il collettore del transistor di uscita, ed I_{IH} quella assorbita dagli ingressi dei ricevitori. Tutte queste correnti, transitando attraverso la resistenza R, non debbono provocare una caduta di tensione tale per cui la tensione sulla linea sia inferiore al valore minimo $V_{OH\ min}$ che può ancora essere considerato come valore logico alto. Ciò permette di determinare il valore massimo, R_{MAX}, ammesso per la resistenza di polarizzazione:

$$V_{CC} - R_{MAX}\,(NI_{OH} + MI_{IH}) \geqslant V_{OH\ min}$$

e cioè

$$R_{MAX} \leqslant \frac{V_{CC} - V_{OH\ min}}{(NI_{OH} + MI_{IH})}$$

Per assicurare che un trasmettitore possa imporre alla linea il valore logico basso, si possono distinguere due casi; nel primo si suppone che sia già stato scelto un tipo di trasmettitore, e quindi se ne conoscano le caratteristiche di uscita, in particolare la massima corrente I_{OMAX} che esso può assorbire, pur assicurando il valore logico basso. In questo caso si deve determinare quale è il minimo valore, R_{min}, che può avere la resistenza R di polarizzazione. Evidentemente l'unico trasmettitore attivo deve poter imporre alla linea il valore logico basso pur assorbendo le correnti di ingresso dei ricevitori I_{IL} e quella proveniente dalla tensione di polarizzazione attraverso la resistenza R:

$$I_{CMAX} \geqslant MI_{IL} + \frac{V_{CC} - V_{OLMAX}}{R_{min}}$$

e cioè

$$R_{min} \geqslant \frac{V_{CC} - V_{OLMAX}}{I_{CMAX} - MI_{IL}}$$

Nel secondo caso, si fa l'ipotesi che sia stato prefissato, per altri motivi (ad esempio, di velocità), il valore R della resistenza di polarizzazione. La precedente relazione permette allora di determinare quale I_{CMAX} debba avere ogni trasmettitore e quindi se ne impongono le caratteristiche di uscita.

1.7.2. *Caratteristiche dinamiche*

Quanto finora visto serve a garantire che il funzionamento in regime statico risulti corretto.
Devono essere soddisfatte determinate condizioni anche durante la commutazione dei valori logici per ottenere un funzionamento privo di errori. Si è già visto infatti che i segnali sui vari bus assumono ben precisi significati solamente in determinati e stabiliti intervalli di tempo. E' necessario quindi che i trasmettitori collegati al bus presentino i valori logici che interessano entro quei precisi intervalli temporali.
Questo comporta che il cambiamento da un livello logico al suo complementare, che ovviamente non può mai avvenire in un tempo nullo, deve presentare delle ben precise caratteristiche di durata.
Il carico visto da un trasmettitore si può considerare equivalente ad una capacità C; in tali ipotesi semplificative la commutazione del livello logico avviene con legge esponenziale e con costante di tempo pari al prodotto di C per la resistenza di uscita del trasmettitore. All'aumentare di tale prodotto cresce il tempo necessario perché la tensione arrivi al valore finale, come indicato nella fig. 1.23 nel caso di una commutazione dal valore logico basso a quello alto.

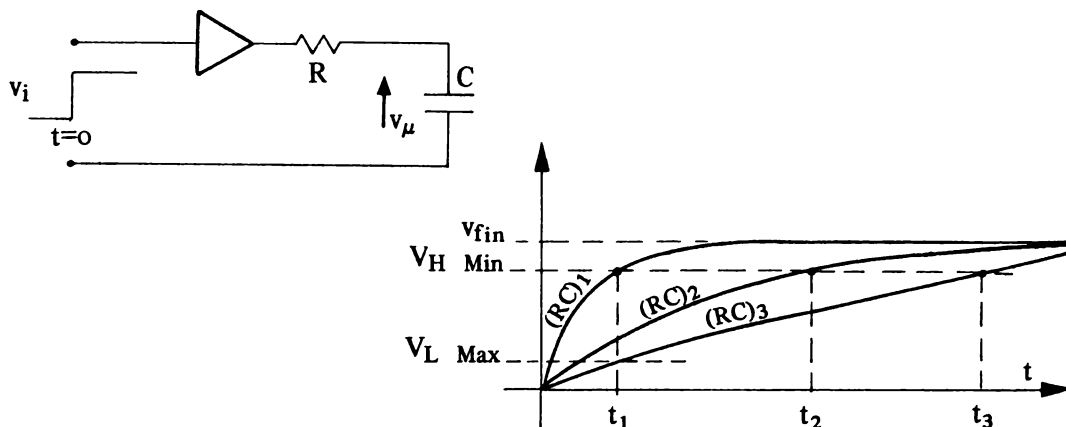



Fig. 1.23
Incremento del tempo di ritardo all'aumentare della costante di tempo RC
$[(RC)_1 < (RC)_2 < (RC)_3]$

Se tale tempo supera certi limiti si può verificare che la tensione sulla linea non abbia raggiunto il suo valore corretto quando un ricevitore va ad acquisirla e ciò provoca errori nella ricezione. Per ovviare a questo inconveniente si può aumentare il tempo che intercorre dall'istante di commutazione a quello nel quale il ricevitore va a prelevare il dato, ma ciò riduce la velocità di funzionamento dell'intero sistema e quindi le sue prestazioni. Per tener conto di queste eventualità il costruttore fornisce sempre le condizioni di carico cui si riferiscono le relazioni temporali che intercorrono tra i segnali di ingresso e di uscita di un dispositivo. Ad esempio nel caso del microprocessore Z80 il tempo di ritardo con cui si stabilizza una linea del bus degli indirizzi è fornito nell'ipotesi che quest'ultima presenti un carico di tipo capacitivo del valore di 100 pF. Tale capacità di carico è costituita nel caso reale dalla capacità dei diversi dispositivi che sono collegati a quella linea, dalla capacità propria della linea stessa ed inoltre da quelle parassite, sempre presenti.
E' poco probabile che la capacità di carico complessiva risulti coincidente con quella indicata dal costruttore; si hanno quindi, nel caso reale, delle variazioni dei tempi di ritardo rispetto a quanto indicato nelle specifiche. Il costruttore fornisce però delle ulteriori indicazioni, dalle quali si ricava come procedere ad una correzione.
Sempre con riferimento allo Z80 sono fornite le seguenti relazioni: se rispetto a quanto indicato si riscontra un aumento della capacità di carico pari a 100 pF i tempi indicati vanno aumentati di 30 ns, mentre se si ha una capacità inferiore a quella indicata di un valore di 50 pF, i tempi vanno diminuiti di 15 ns. Ammettendo che nell'intervallo citato la variazione dei tempi sia lineare con la variazione della capacità, si può dire allora che si ha una variazione di 30/100 = 0,3 ns per una variazione di un pF. In questo caso il coefficiente di variazione è uguale sia per aumento che diminuzione della capacità; in altri invece tale coefficiente è maggiore per valori più grandi di capacità ed inferiore per valori più piccoli. Si noti comunque che la variazione ammessa per le capacità di carico è di solito limitata.
Per calcolare le variazioni dei tempi di ritardo in un caso reale si consideri il seguente esempio: tutte le porte riceventi collegate ad una linea di un bus siano costruite con tecnologia NMOS per cui la corrente di ingresso assume il valore tipico di $\pm$ 10 μA sia al livello alto che basso: I_{IL} = = $-10\ \mu$A; $I_{IH} = +\ 10\ \mu$A.
Il microprocessore Z80 può allora pilotare nello stato alto 250/10 = 25 porte, mentre nello stato basso potrebbe pilotarne addirittura 180. Se ci si limita all'analisi in regime statico si potrebbe concludere che si potrebbe pilotare ben 25 porte NMOS. La situazione è diversa in regime dinamico: infatti una porta NMOS presenta tipicamente una capacità di ingresso di 10 pF, per cui, se fossero effettivamente collegate 25 porte, il

carico presenterebbe una capacità di 250 pF, che é superiore a quella di 100 pF cui si riferiscono le diverse relazioni temporali. Si noti che a questa capacità dovrebbe essere ulteriormente aggiunta anche la capacità propria del bus e le capacità parassite con conseguente deterioramento del funzionamento.
In questo caso si può risolvere l'inconveniente mediante l'inserzione di un buffer il quale, oltre a garantire il pilotaggio corretto delle linee in regime statico, permetta altresì di mantenere elevata la velocità di commutazione. Nell'esempio precedente, il microprocessore Z80, se caricato con 250 pF, stabilizzerà la linea pilotata in un tempo al massimo pari a:

$$160 \text{ ns} + 45 \text{ ns} = 205 \text{ ns}$$

cioè in un tempo maggiore del 25% circa, avendo trascurato per semplicità le capacità della linea del bus e quelle parassite.
Se tale tempo di stabilizzazione è eccessivo si può interporre tra un piedino di uscita dello Z80 e la linea del bus un buffer con le seguenti caratteristiche:

$C_{IN} = 50$ pF;

$I_{IN} = \pm 80\ \mu A$;

tempo di ritardo ingresso-uscita: 30 ns;

capacità di carico: 300 pF;

$I_{OL} = I_{OH} = 50$ mA;

si avrebbe ancora il corretto pilotaggio in regime statico e per quanto riguarda quello dinamico un ritardo massimo pari a:

$$160 \text{ ns} - 15 + 30 = 175 \text{ ns}$$

ove i − 15 ns derivano dal fatto che, nelle solite ipotesi, il carico capacitivo visto dallo Z80 è diminuito rispetto a quello indicato nelle specifiche. Con questa .soluzione l'aumento del tempo di stabilizzazione della linea di indirizzo è solo del 10% circa.
Un problema che può talvolta presentarsi è quello che i carichi collegati alla linea del bus siano superiori a quelli pilotabili da un qualsiasi buffer: in questo caso è ammessa l'utilizzazione di due o più buffer in parallelo. Un altro caso che si presenta in pratica è quello di una lunghezza eccessiva del bus, per cui potrebbe rendersi necessaria la suddivisione dello

stesso in due o più tratti collegati tra loro da buffer che fungono da ripetitori; nell'ambito di un singolo tratto rimangono valide le considerazioni fatte.
In entrambi i due casi su esposti il comando di attivazione dei buffer deve essere opportunamente organizzato per evitare conflitti sulla linea.

1.8. Standardizzazione del bus dei microelaboratori

I vari blocchi funzionali che costituiscono un microelaboratore sono di solito realizzati su schede separate in ognuna delle quali sono presenti un certo numero di dispositivi per adempiere a determinate funzioni; si trovano ad esempio in commercio delle schede in cui sono presenti il microprocessore con il generatore di clock e l'hardware relativo alla gestione delle interruzioni, altre in cui sono contenuti un certo numero di circuiti integrati che realizzano un banco di memorie di programma, ecc.
Le diverse schede sono collegate tra loro tramite un bus comune, detto bus di sistema, che frequentemente è realizzato su un unico supporto fisico in cui sono allocate le linee che lo costituiscono e dove sono presenti dei connettori che permettono l'inserzione delle schede stesse. Queste ultime debbono presentare allora sui rispettivi connettori i vari segnali in modo ordinato, e cioè ad ogni posto del connettore deve essere assegnato, qualsiasi sia il tipo di scheda, un segnale che ha un significato ed una funzione prestabiliti nell'ambito di quel bus.
Parecchi sono stati i tentativi fatti per raggiungere una standardizzazione ed effettivamente si sono imposti un numero peraltro limitato di bus standard ai quali i costruttori di schede si attengono.
Per una efficiente standardizzazione è necessario siano rispettati sia dei requisiti fisici che funzionali.
L'aspetto fisico, anche se molto importante ai fini pratici, è concettualmente di facile soluzione; stabilito ad esempio che in un insieme ordinato di 50 linee il bus dei dati è formato dalle line L22, L23, ..., L29, si deve anche stabilire che nella linea L22 si trasmette il bit meno significativo, D0, del dato, nella linea L23 quello immediatamente più significativo e così via. E' ovvia l'importanza di fissare tale convenzione, ma è altrettanto ovvio il modo di procedere per arrivare ad una soluzione. In fase realizzativa si tratta poi di rispettare le convenzioni adottate.
L'aspetto funzionale presenta invece delle difficoltà notevolmente maggiori. Si deve tener presente che per fissare una standardizzazione è necessario che questa possa essere utilizzata in applicazioni anche notevolmente diverse e quindi in elaboratori che impiegano tipi diversi di microprocessori, come pure differenti unità periferiche.
Per quanto riguarda il bus dei dati e quello degli indirizzi si tratta solamente di fissare il massimo numero di bit che possono essere contemporaneamente trasferiti in parallelo.

Nel bus dei dati si deve stabilire se la standardizzazione deve essere valida solo per microprocessori ad 8 bit o se deve valere anche per quelli a 16 bit: nel primo caso il bus dei dati è formato da 8 linee, nel secondo caso da 16. Analogamente si procede per il bus degli indirizzi.
Tutto sommato anche la standardizzazione funzionale dei bus dei dati e degli indirizzi non presenta problemi.
Dove sorgono invece notevoli difficoltà è nel bus di controllo: si devono infatti stabilire quali sono le funzioni di controllo in un elaboratore e le modalità con cui è conveniente applicarle.
Ad esempio, nel bus di controllo è utilizzata una linea attraverso la quale i vari dispositivi presenti nelle schede possono inviare delle richieste di interruzione all'unità centrale.
Quest'ultima deve poi informare i dispositivi periferici quando accetta di servire la richiesta.
Il microprocessore Z80 fornisce l'informazione di accettazione attivando contemporaneamente i due segnali M1/ e IORQ/. A questo punto ci si può chiedere se l'informazione di accettazione della richiesta di interruzione è più conveniente sia trasferita attraverso il bus mediante una sola linea o in forma codificata su più linee attraverso le quali possono venire trasferite anche altre informazioni.
Per la realizzazione di un microelaboratore che utilizza solamente lo Z80 e componenti periferici della stessa famiglia è più conveniente la seconda soluzione in quanto in caso contrario si deve prevedere sia un circuito che generi il segnale di accettazione, ACK, da inviare sul bus sia un altro circuito che deve procedere all'operazione inversa e cioè che dal segnale di accettazione possa ricavare IORQ/ e M1/, in quanto i dispositivi della famiglia Z80, come si vedrà, acquisiscono l'accettazione della interruzione da parte della CPU solamente attraverso questi due segnali. In fig. 1.24 sono riportati gli schemi che si ottengono adottando le due soluzioni.
La prima soluzione (a) va bene solamente per microelaboratori che utilizzano componenti della famiglia Z80, mentre una standardizzazione deve avere validità più generale, anche se ciò comporta la realizzazione di circuiti aggiuntivi.
Per il bus di controllo si tratta allora innanzi tutto di stabilire quali debbano essere le informazioni da esso fornite; successivamente si stabiliscono le modalità con cui tali informazioni debbono essere fornite e solo a questo punto ci si può occupare dell'aspetto fisico di tale bus.
Le principali informazioni che devono essere fornite da un bus di controllo riguardano:

- l'indicazione delle operazioni da eseguire sulla memoria o sugli organi periferici; in particolare il sistema deve essere in grado di distinguere quattro fondamentali tipi di operazioni: lettura e scrittura della memoria e lettura e scrittura di organi periferici;

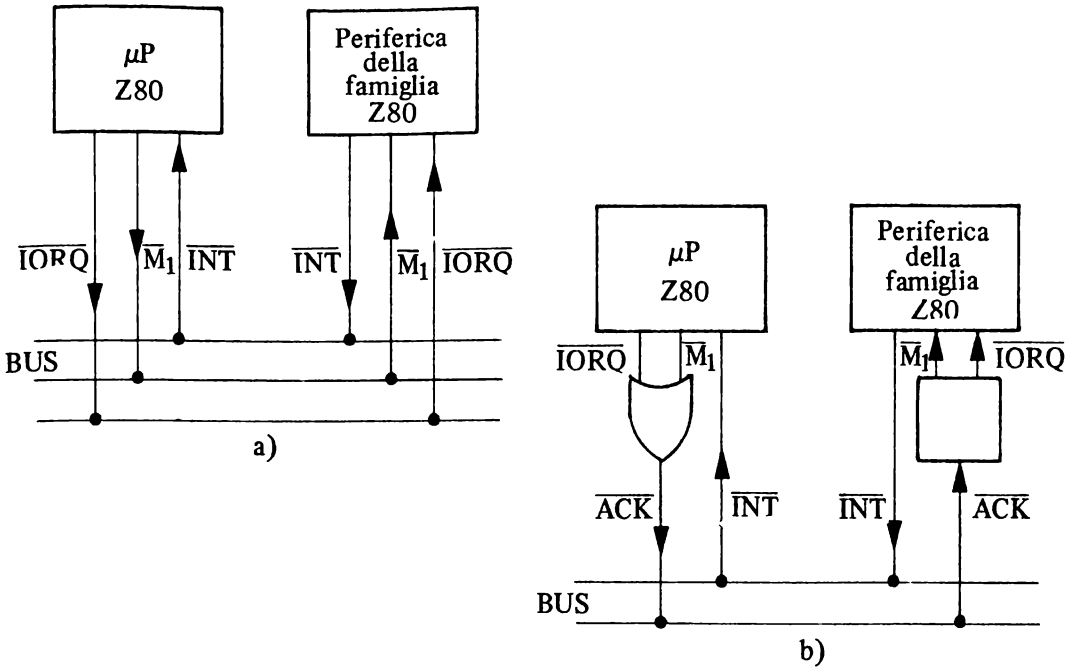



Fig. 1.24
Differenti organizzazioni di un bus.

- la sincronizzazione dei vari dispositivi collegati al bus: ad esempio segnalazioni di attesa o di pronto ad eseguire una certa azione;
- le richieste ed accettazioni di interruzioni;
- le segnalazioni per la cessione e per l'assunzione della funzione di master.

Sono inoltre presenti delle linee per il segnale di azzeramento iniziale, RESET, che permette di stabilire l'istante di inizio delle attività dei singoli dispositivi, altre linee per vari segnali di clock necessari alla sincronizzazione, ecc.
Per quanto riguarda le modalità con cui il bus di controllo fornisce le informazioni, atte ad individuare una determinata azione, si possono trovare soluzioni diverse che comportano l'utilizzazione di un numero di linee differente da caso a caso.
Si consideri ad esempio l'attività di trasferimento dei dati: si presentano in genere quattro situazioni distinte, a seconda del dispositivo interessato, memoria o periferica, e del tipo di operazione, lettura o scrittura.
Allo scopo potrebbero essere usati due soli segnali: M/IO e R/W. Quando il primo assume il valore logico alto l'azione in atto si riferisce alla memoria, in caso contrario agli organi di ingresso o uscita. Per il secondo, quan-

do esso assume un valore logico alto, ciò sta a significare che si vuole eseguire una operazione di lettura; se invece il valore è basso, l'operazione è di scrittura.
Si potrebbero altresì usare tre segnali: MEM/, IO/, R/W nel qual caso i primi due, quando attivi (si noti che non lo possono essere contemporaneamente) indicano il tipo di dispositivo interessato al trasferimento, mentre il terzo segnale ha lo stesso significato del caso precedente.
Ancora, potrebbero essere utilizzati quattro segnali distinti: MEM/, IO/, RD/, WR/ dei quali i primi due individuano il dispositivo interessato, i restanti l'operazione da effettuare.
Come si può dedurre da quanto esposto le possibilità di organizzare un bus di controllo sono molte ed le varie standardizzazioni attualmente adottate si differenziano principalmente per tale bus.
La convenienza ad adottare un tipo di bus dipende fortemente dal microprocessore che si vuole utilizzare e dall'organizzazione del microelaboratore.
Oltre ai vari segnali appartenenti ai tre tipi di bus è necessario inoltre in un bus di sistema un certo numero di linee per le tensioni di alimentazione che conviene avere in numero abbastanza numeroso, sia come tipo (5V, –5V, 12V, –12V, ecc.) che come numero (più linee a 5V) in modo che la corrente circolante in ognuna di esse sia limitata e quindi contenuta la variazione di tensione nei vari punti. E' sempre conveniente inoltre avere a disposizione delle linee "libere" per poterle utilizzare in applicazioni particolari e rendere più flessibile l'utilizzazione del bus.
I principali tipi di bus standard attualmente in uso sono l'S100, il MULTIBUS, lo Z-BUS, ecc.; per quanto riguarda le loro caratteristiche si rimanda ai manuali forniti dai vari costruttori.

1.9. Bus contention

Nel descrivere le caratteristiche funzionali del bus in un microelaboratore si è più volte sottolineata la necessità di evitare che due o più trasmettitori siano attivi contemporaneamente. Ciò comporta infatti la presenza nel bus interessato di valori logici indeterminati, con conseguente perdita della informazione trasmessa, e con la possibilità di distruzione dei trasmettitori.
Tale conflittualità nel bus (bus contention) può verificarsi per vari motivi in genere imputabili alla finita e non nulla velocità di risposta di un dispositivo ai segnali di controllo.
Quest'ultima comporta che un dispositivo che assume il ruolo di trasmettitore, e quindi attiva le sue uscite dopo che ha ricevuto il segnale di abilitazione, al cessare di questo in genere passa nello stato di alta impedenza entro un certo tempo, detto tempo di hold.
A causa dei vari ritardi di propagazione può succedere che un trasmettitore sia attivato prima che il trasmettitore precedentemente attivo passi nello stato ad alta impedenza.

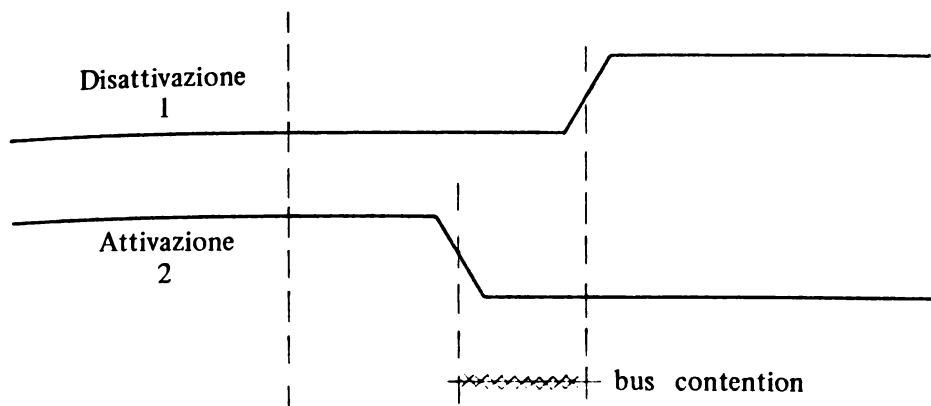



Fig. 1.25.
Situazione in cui si verifica il bus contention.

+V_{CC} +V_{CC}

T_1 T_3

1 ∅

T_2 T_4

?

trasmettitore 1 trasmettitore 2

linea del bus

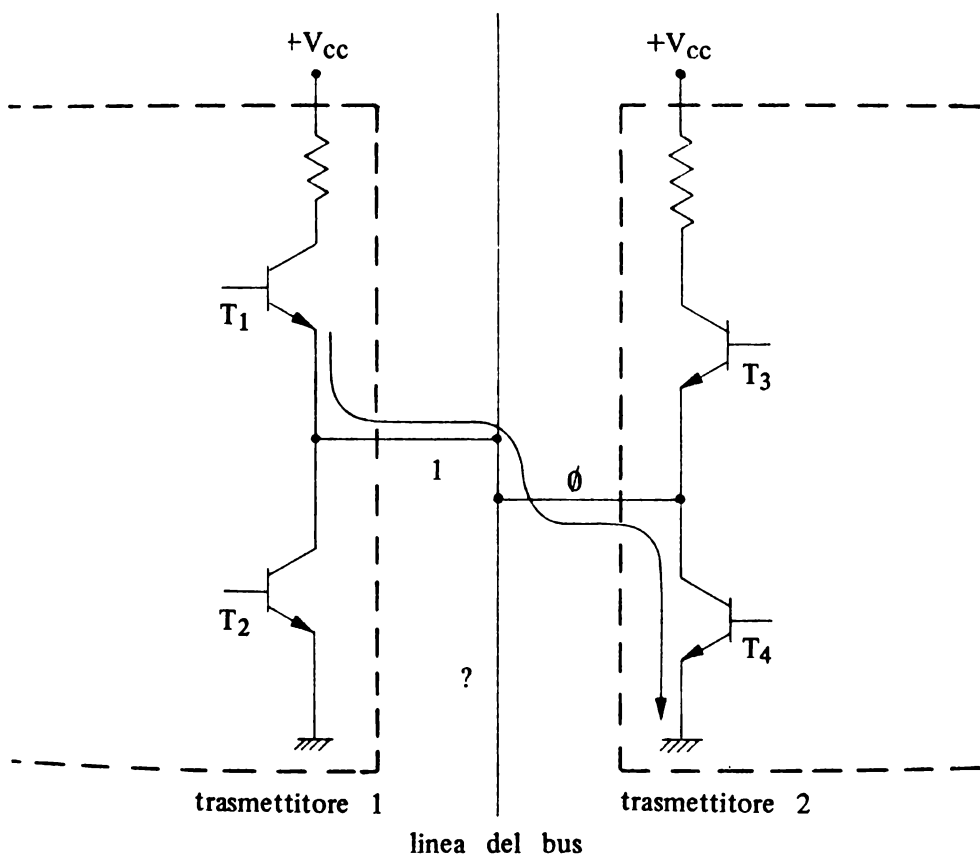



Fig. 1.26
Condizione di bus contention quando sono attivi 2 trasmettitori.

Ciò provoca inevitabilmente un fenomeno di bus contention come è illustrato nella fig. 1.25.
In genere il fenomeno si esaurisce in tempi brevi, ma può provocare una circolazione elevata di corrente dato che, su una medesima linea del bus, uno dei trasmettitori può voler imporre un valore logico alto e l'altro un valore logico basso. Tale situazione è rappresentata in forma schematica in fig. 1.26.
Gli impulsi di corrente così generati, oltre a poter distruggere qualche volta gli stadi di uscita dei due trasmettitori, possono indurre dei disturbi nel sistema e provocare degli errori di funzionamento su altri dispositivi collegati al bus e non interessati alla trasmissione, anche nel caso in cui il ricevitore acquisisca il dato presente sul bus quando il bus contention si è esaurito.
Queste anomalie sono difficilmente individuabili e perciò devono essere evitate in modo assoluto.
Più grave è il fenomeno di bus contention che si verifica per tempi lunghi: in questo caso, oltre agli inconvenienti su accennati, c'è una concreta possibilità di distruzione dei trasmettitori.
In seguito saranno illustrati degli schemi circuitali in cui può avvenire il bus contention ed i metodi per evitare tale fenomeno.

Capitolo 2

MEMORIE STATICHE

2.1. Dispositivi di memoria

In ogni elaboratore elettronico, come indicato nello schema a blocchi di fig. 1.1, sono sempre presenti dei dispositivi di memoria nei quali si conservano i risultati intermedi e finali delle elaborazioni eseguite dall'unità centale, oltre ai programmi con cui si predispongono le elaborazioni desiderate.
I dispositivi di memoria attualmente impiegati possono essere realizzati con materiali e tecnologie diverse che conferiscono loro particolari caratteristiche.
Una prima classificazione delle memorie permette di dividerle in due gruppi fondamentali: le memorie volatili e quelle non volatili.
Al primo gruppo appartengono i dispositivi di memoria che non sono in grado di conservare l'informazione in essi immagazzinata quando viene a mancare la tensione di alimentazione; sono in genere utilizzati come memorie temporanee per i risultati parziali.
Le memorie non volatili invece conservano l'informazione indipendentemente dalla presenza o meno della tensione di alimentazione.
Una tipica applicazione è quella della realizzazione di archivi e della memorizzazione di programmi.
Alla prima classe appartengono i vari tipi delle cosidette memorie RAM (Random access memory), mentre alla seconda appartengono le memorie magnetiche a dischi, a nastri, a nuclei di ferrite, o a bolle magnetiche e le memorie a semiconduttore a sola lettura (ROM: ready only memory).
Le memorie magnetiche sono in grado di memorizzare un gran numero di dati e generalmente sono utilizzate come memorie di massa e per la formazione di archivi di dati. Le memorie a nuclei di ferrite, un tempo molto impiegate, trovano ora applicazioni solo in casi particolari. Quelle a bolle magnetiche, da poco tempo disponibili sul mercato, cominciano ad essere utilizzate nei sistemi a microprocessore e se ne prevede una utilizzazione via via crescente.

Le memorie impiegate nei microelaboratori sono quasi esclusivamente a semiconduttore e solo tali tipi saranno qui presi in considerazione.
Le memorie RAM possono essere di tipo statico e dinamico. Una memoria statica è in grado di conservare l'informazione in essa immagazzinata finché si mantiene la tensione di alimentazione. Nelle memorie RAM dinamiche invece l'informazione viene mantenuta solo per un breve intervallo di tempo, dopo il quale, se non viene ripristinata, va persa anche se è presente la tensione di alimentazione. L'operazione di ripristino prende il nome di rinfresco della memoria e per la sua esecuzione è necessario predisporre un adatto circuito esterno, a volte pilotato dall'unità centrale. Tali tipi di memorie sono realizzate con tecniche costruttive che consentono un maggior grado di integrazione rispetto a quelle statiche e quindi una maggiore capacità di memoria a parità di dimensioni della piastrina di materiale semiconduttore usata, il che si traduce in un costo inferiore.
Le memorie ROM possono a loro volta essere suddivise in diversi tipi. Esistono infatti le ROM propriamente dette fornite direttamente dal costruttore con immagazzinate le informazioni necessarie ad una particolare applicazione. A questo tipo appartengono ad esempio le ROM utilizzate come tabelle per il calcolo di funzioni trigonometriche, quelle utilizzate nei generatori di caratteri per tubi a raggi catodici o display a matrice di punti, ecc.
E' anche possibile chiedere al costruttore di ROM di fornire le stesse con immagazzinate delle informazioni specificate dall'acquirente per una particolare applicazione: in questo caso il costo è ragionevole solo per ordinazioni dell'ordine delle decine di migliaia.
Un altro tipo di memoria ROM prende il nome di PROM (Programmable Read Only Memory): in essa è l'utilizzatore stesso che memorizza le informazioni desiderate con una adatta apparecchiatura. Tale operazione può essere fatta una sola volta; la loro tipica applicazione è nella produzione di piccole serie di apparecchiature.
Per la realizzazione di prototipi è invece generalmente utilizzata una ROM di tipo EPROM (Erasable Programmable Read Only Memory); in essa è possibile sia la memorizzazione di informazioni da parte dell'utilizzatore ed anche la eventuale cancellazione delle stesse, utilizzando raggi ultravioletti; queste operazioni possono essere eseguite più volte.
Esiste un altro tipo di ROM: le EAROM (Electrically Alterable ROM) o EEROM (Electrically Erasable ROM). Su esse è possibile effettuare la cancellazione utilizzando solo segnali elettrici, senza dover ricorrere ai raggi ultravioletti. Potrebbero essere considerate delle memorie sia di lettura che di scrittura, ma si deve tener presente che l'operazione di scrittura è particolarmente lenta rispetto a quella di lettura. Sono l'ultimo tipo di ROM apparso sul mercato; presentano attualmente un costo elevato ma se ne prevede nel prossimo futuro una notevole diffusione.

In uno qualsiasi dei tipi di memoria a semiconduttore finora esaminate gli elementi che servono alla memorizzazione, o celle di memoria, sono generalmente disposti all'interno del circuito integrato secondo una configurazione a matrice, di m righe ed n colonne, dove m ed n sono numeri interi potenze di due. In ognuna delle singole celle è possibile la memorizzazione di un bit di informazione. Le capacità di memoria, cioè il numero di bit disponibili, sono quindi potenze di due e possono essere ad esempio 256, 512, 1024,... per arrivare anche a 128 o 256 Kbit.
Non sempre si può accedere al singolo bit: in parecchi casi è infatti possibile l'accesso a 4 oppure a 8 bit a seconda di come è organizzata la memoria; ad esempio in una memoria di 4096 bit possono essere presenti o 4096 locazioni da 1 bit, o 1024 da 4 bit, o 512 da 8 bit.

2.2. Organizzazione interna di una memoria

I segnali che interessano una generica memoria sono indicati in fig. 2.1:

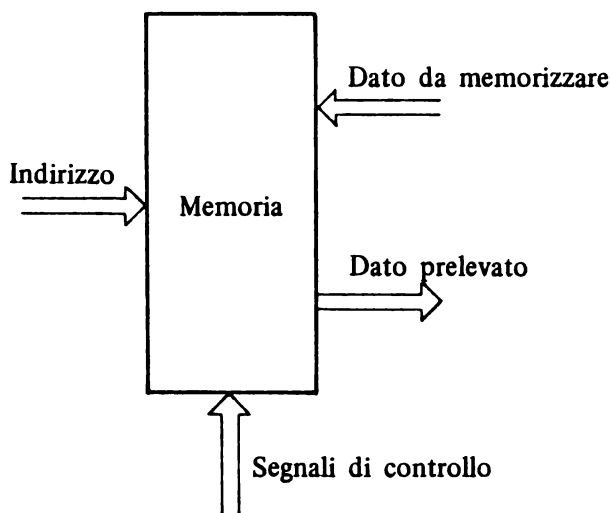



Fig. 2.1
Segnali in una memoria generica.

I segnali di indirizzo permettorno di individuare in modo univoco quali celle di memoria sono interessate alle operazioni di lettura o scrittura.
Le linee dei dati, in scrittura ed in lettura, possono coincidere oppure essere distinte; ovviamente nel caso di una memoria ROM non sono presenti quelle per l'invio dei dati da scrivere.

I segnali di controllo permettono di fornire al dispositivo quelle informazioni necessarie ad un corretto funzionamento: essi indicano quando devono essere considerati validi i segnali di indirizzo, quale operazione si desidera eseguire, ecc. Si noti la presenza di tre tipi di informazioni: indirizzi, dati e controlli, come già riscontrato nei bus.
Esistono diversi modi per ottenere dall'indirizzo la selezione della locazione desiderata; in fig. 2.2 è schematizzato il caso di una memoria ROM con capacità di 1024 bit organizzati in 256 locazioni di 4 bit ciascuna.

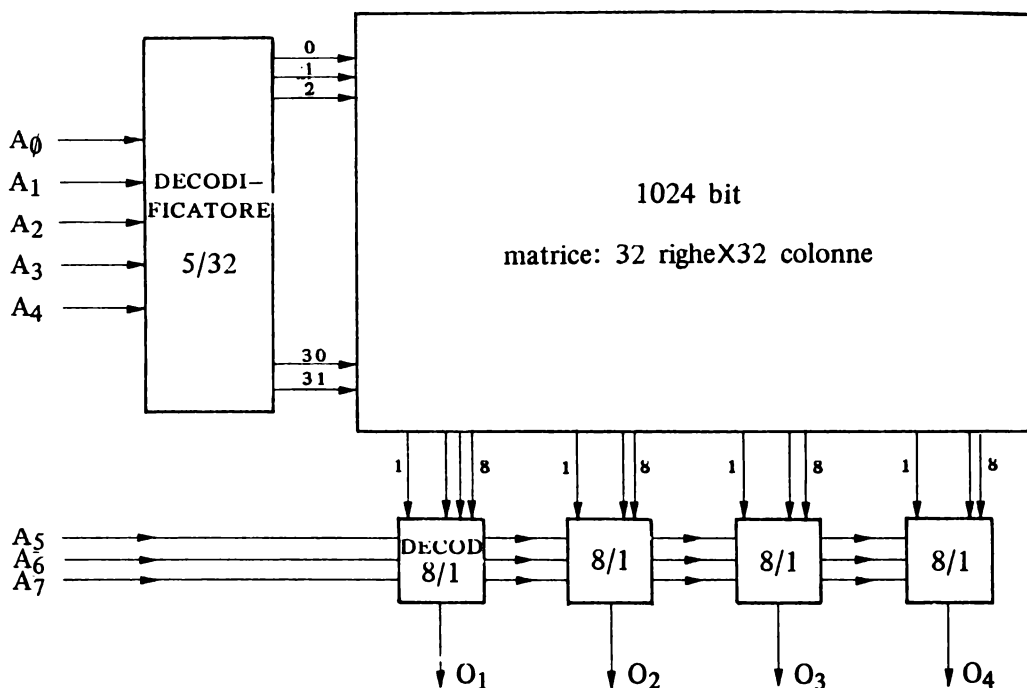



Fig. 2.2
Schema a blocchi di una memoria ROM da 256 x 4 bit.

Le 1024 celle della memoria sono disposte in una matrice di 32 righe e di 32 colonne; mediante i cinque bit meno significativi dell'indirizzo un decodificatore interno attiva una ed una sola delle 32 righe della matrice, quella individuata cioè dalla configurazione presente sulle suddette cinque linee di indirizzo.
Sulle 32 uscite dalla matrice saranno presenti allora i valori memorizzati nelle celle della riga selezionata; tali uscite sono ordinatamente inviate a quattro circuiti multiplexer, ognuno ad 8 ingressi.

Ogni multiplexer fornisce in uscita il valore presente su quello degli 8 ingressi individuato dal valore assunto da tre segnali di selezione, che in questo caso sono costituiti dai restanti bit dell'indirizzo: allora alle uscite dei quattro multiplexer, uscite che coincidono con quelle dei dati della memoria, sono presenti i valori memorizzati nella locazione individuata dai valori assunti dalle otto linee di indirizzo.
Una volta scelta l'organizzazione interna, restano fissati e il numero di bit necessario per l'indirizzo e il numero di bit che si possono leggere (o scrivere) contemporaneamente.
I segnali di controllo necessari al funzionamento variano al variare del tipo di memoria e del costruttore.
Le informazioni inviate mediante i segnali di controllo sono di due tipi: il primo individua l'operazione, di lettura o di scrittura, che si vuole eseguire sulla memoria; il secondo è utilizzato in genere per sincronizzare il funzionamento della memoria mettendola in attività solamente durante certi intervalli di tempo e solo quando sono valide le informazioni presenti sulle linee degli indirizzi e dei dati.
Queste informazioni debbono essere fornite con modalità diverse e tramite un certo numero di segnali per cui sono in genere necessari dei circuiti di adattamento che generino i segnali richiesti in funzione di quelli messi a disposizione dall'unità centrale.
Per quanto riguarda le linee dei dati in uscita, queste presentano di solito caratteristiche three-state per permettere un facile collegamento ad un bus.

2.3. Caratteristiche elettriche di una memoria

Le case costruttrici di memorie forniscono le caratteristiche elettriche del dispositivo relative sia al regime statico che a quello dinamico; in quest'ultimo caso sono anche specificate le relazioni temporali fra i diversi segnali.
Tra le caratteristiche statiche sono generalmente indicate:

- il campo di valori ammessi per la o le tensioni di alimentazione;
- le correnti tipiche di alimentazione, di solito per i valori di tensione nominale;
- i valori di tensione di ingresso che sono considerati a valore logico alto o basso e le correnti assorbite o erogate dagli ingressi a tali livelli di tensione;
- analoghi valori per le tensioni e le correnti di uscita;
- i valori di corrente erogati o assorbiti con l'uscita nello stato di alta impedenza.

Le caratteristiche dinamiche indicano i legami temporali fra i diversi segnali in gioco e forniscono informazioni sulla sequenza con cui gli ingressi del-

la memoria devono essere attivati per ottenere le operazioni desiderate.
Per illustrare tali caratteristiche si prenda in considerazione una tipica memoria RAM statica, la 2102AL-2, la quale ha una capacità di 1024 x 1 bit, cioè ognuno dei bit è singolarmente indirizzabile. Lo schema a blocchi, i segnali presenti ai vari piedini e la tabella di verità sono riportati in fig. 2.3. Le varie celle che la compongono sono organizzate in una matrice di 32 righe per 32 colonne e sono necessarie perciò 10 linee di indirizzo per individuare uno dei 1024 bit. Sono previste due linee separate, Data Input e Data Output, rispettivamente per l'ingresso e per l'uscita dei dati, e due segnali di controllo che consentono di attivare il circuito di memoria (CE/) e di indicare il tipo di operazione da eseguire (R/W).

D_{IN}	DATA INPUT
A_0-A_9	ADDRESS INPUTS
R/W	READ/WRITE INPUT
$\overline{CE}$	CHIP ENABLE
D_{OUT}	DATA OUTPUT
V_{CC}	POWER (+5V)

$\overline{CE}$	R/W	D_{IN}	D_{OUT}	MODE
H	X	X	HIGH Z	NOT SELECTED
L	L	L	L	WRITE "0"
L	L	H	H	WRITE "1"
L	H	X	D_{OUT}	READ

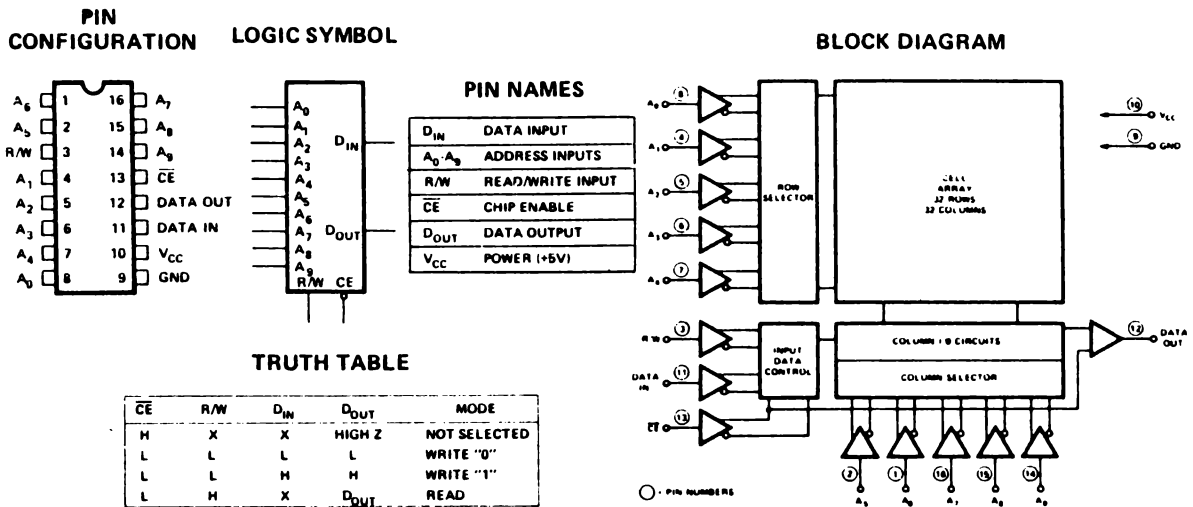



Fig. 2.3
Memoria RAM statica 2102 (1 K x 1): schema a blocchi.
(Intel, Component Data Catalog 1979).

La tabella di verità indica come questi due ultimi segnali sono utilizzati: come si può notare con il segnale CE/ a valore logico alto il dispositivo non è attivo e l'uscita si trova nello stato di alta impedenza. Solo con il segnale CE/ a livello logico basso sul dispositivo si possono effettuare operazioni di lettura e/o scrittura. Se il segnale R/W si trova a livello alto significa che si vuole eseguire una operazione di lettura: il bit letto sarà fornito sul piedino D_{OUT}. Se invece R/W si trova a livello basso si tratta di una

operazione di scrittura del valore presente sul piedino D_{IN}.
Le caratteristiche per il funzionamento statico sono compatibili con quelle di una normale porta TTL, in particolare per quanto riguarda le correnti assorbite o erogate e sono riportate in fig. 2.4.

Absolute Maximum Ratings*

Ambient Temperature Under Bias	-10°C to 80°C
Storage Temperature	–65°C to +150°C
Voltage On Any Pin With Respect To Ground	–0.5V to +7V
Power Dissipation	1 Watt

*COMMENT:
Stresses above those listed under "Absolute Maximum Rating" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or at any other condition above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

D. C. and Operating Characteristics

T_A = 0°C to 70°C, V_{CC} = 5V ±5% unless otherwise specified.

Symbol	Parameter	2102A, 2102A-4 2102AL, 2102AL-4 Limits Min.	Typ.[1]	Max.	2102A-2, 2102AL-2 Limits Min.	Typ.[1]	Max.	Unit	Test Conditions
I_{LI}	Input Load Current		1	10		1	10	μA	V_{IN} = 0 to 5.25V
I_{LOH}	Output Leakage Current		1	5		1	5	μA	$\overline{CE}$ = 2.0V, V_{OUT} = V_{OH}
I_{LOL}	Output Leakage Current		-1	-10		-1	-10	μA	$\overline{CE}$ = 2.0V, V_{OUT} = 0.4V
I_{CC}	Power Supply Current		33	Note 2		45	65	mA	All Inputs = 5.25V, Data Out Open, T_A = 0°C
V_{IL}	Input Low Voltage	-0.5		0.8	-0.5		0.8	V	
V_{IH}	Input High Voltage	2.0		V_{CC}	2.0		V_{CC}	V	
V_{OL}	Output Low Voltage			0.4			0.4	V	I_{OL} = 2.1mA
V_{OH}	Output High Voltage	2.4			2.4			V	I_{OH} = -100μA

Notes: 1. Typical values are for T_A = 25°C and nominal supply voltage.
2. The maximum I_{CC} value is 55mA for the 2102A and 2102A-4, and 33mA for the 2102AL and 2102AL-4.

Fig. 2.4
Caratteristiche statiche della memoria RAM statica 2102. (Intel Component Data Catalog 1979)

2.3.1. *Ciclo di lettura*

L'operazione di lettura si attua in tre fasi:

- invio dell'indirizzo che seleziona il bit interessato;
- attivazione della memoria;
- prelievo del bit all'uscita della memoria.

In fig. 2.5 è riportato l'andamento temporale dei vari segnali interessati con i valori limite dei tempi impiegati nelle diverse fasi.

A. C. Characteristics T_A = 0°C to 70°C, V_{CC} = 5V ±5% unless otherwise specified

READ CYCLE

Symbol	Parameter	2102A-2, 2102AL-2 Limits (ns) Min.	Max.	2102A, 2102AL Limits (ns) Min.	Max.	2102A-4, 2102AL-4 Limits (ns) Min.	Max.
t_{RC}	Read Cycle	250		350		450	
t_A	Access Time		250		350		450
t_{CO}	Chip Enable to Output Time		130		180		230
t_{OH1}	Previous Read Data Valid with Respect to Address	40		40		40	
t_{OH2}	Previous Read Data Valid with Respect to Chip Enable	0		0		0	

READ CYCLE

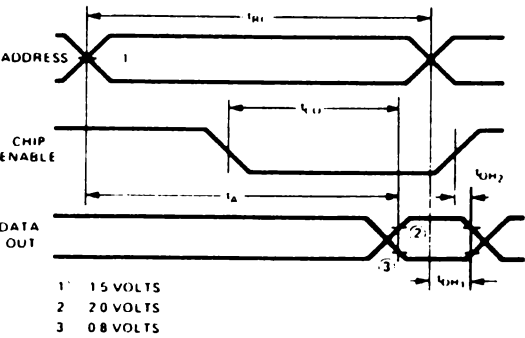



Fig. 2.5
Ciclo di lettura della memoria RAM statica 2102. (Intel Component Data Catalog 1979)

Come si può notare, non è riportato il segnale R/W che permette di indicare il tipo di operazione da eseguire; si deve intendere che tale segnale sia

a livello alto (lettura) e che tale livello sia anche quello di riposo.
L'operazione di lettura inizia quando si invia l'indirizzo del bit desiderato e termina quando è possibile inviare un nuovo indirizzo per eseguire una successiva operazione sulla memoria. Tale intervallo di tempo prende il nome di tempo per un ciclo di lettura (t_{RC}) e dura al minimo 250 ns per la 2102AL-2.
Si intende invece come tempo di accesso, t_A, il tempo che intercorre dall'istante di cui è stato fornito l'indirizzo all'istante in cui il bit richiesto è disponibile sul piedino di uscita; è necessario ovviamente che sia attivo il segnale di abilitazione, CE/, ed anzi occorre che esso sia stato attivato almeno da 130 ns perché l'uscita sia valida.
Quest'ultimo tempo è inferiore a quello di accesso e questo a sua volta lo è riguardo al tempo necessario ad un ciclo di lettura ($t_{CO} < t_A < t_{RC}$).
Il dato in uscita resta valido fino all'istante in cui sono cambiati i segnali di indirizzo, oppure è disattivato il segnale CE/. Nel primo caso il dato si mantiene valido ancora per 40 ns dopo la variazione degli indirizzi (t_{OH1}); nel caso in cui si disattivi CS/ la validità del dato viene a cessare immediatamente ($t_{OH2} = 0$).
Quanto è stato esemplificato è abbastanza tipico di come si esegue un ciclo di lettura: al variare del tipo di memoria si possono trovare delle differenze che riguardano i vari tempi in gioco ed i segnali di abilitazione, CE/, i quali possono essere più di uno; è spesso presente anche un segnale OD, (OUTPUT DISABLE), la cui attivazione fa assumere all'uscita lo stato di three-state, anche se il dato richiesto è stato prelevato dalla matrice e portato sul buffer di uscita interno alla memoria stessa.

2.3.2. *Ciclo di scrittura*

Per quanto riguarda l'operazione di scrittura anch'essa si articola in tre fasi distinte:

- invio dell'indirizzo relativo alla cella di memoria interessata;
- attivazione della memoria con indicazione dell'operazione da eseguire;
- invio del dato da memorizzare.

I diagrammi temporali relativi a tale ciclo sono riportati nella fig. 2.6.
L'operazione di scrittura inizia con l'invio dell'indirizzo ed ha termine quando è possibile inviare un nuovo indirizzo per eseguire un'altra operazione sulla memoria; questo tempo di ciclo di scrittura, t_{WC}, nel caso in esame vale 250 ns al minimo.
Debbono essere attivati quindi i due segnali CE/ e R/W, portandoli entrambi al livello logico basso; in particolare il segnale R/W va attivato con un certo ritardo dopo che sono stati forniti gli indirizzi ($t_{AW} \underset{min}{\geqslant} 20$ ns) e deve avere una durata minima pari a $t_{WP} = 180$ ns. Il segnale CE/ deve avere una durata non inferiore a quella di R/W.

WRITE CYCLE

Symbol	Parameter	2102A-2, 2102AL-2 Limits (ns) Min.	Max.	2102A, 2102AL Limits (ns) Min.	Max.	2102A-4, 2102AL-4 Limits (ns) Min.	Max.
t_{WC}	Write Cycle	250		350		450	
t_{AW}	Address to Write Setup Time	20		20		20	
t_{WP}	Write Pulse Width	180		250		300	
t_{WR}	Write Recovery Time	0		0		0	
t_{DW}	Data Setup Time	180		250		300	
t_{DH}	Data Hold Time	0		0		0	
t_{CW}	Chip Enable to Write Setup Time	180		250		300	

WRITE CYCLE

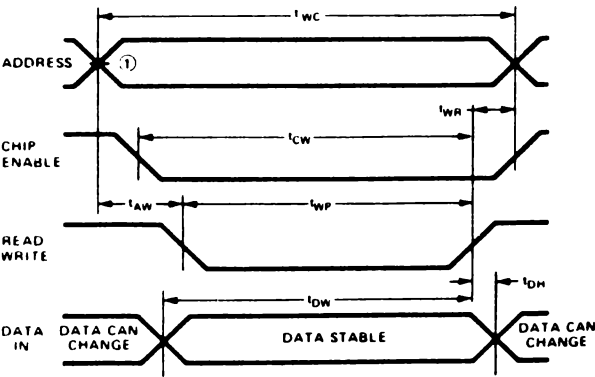



Fig. 2.6
Ciclo di scrittura della memoria RAM statica 2102 (Intel Component Data Catalog 1979)

Il dato da memorizzare deve essere disponibile al piedino di ingresso D_{IN} prima o al massimo contemporaneamente all'attivazione del segnale R/W, e quando quest'ultimo viene disattivato anche il dato può essere contemporaneamente tolto dalla linea di ingresso, come indicato nella tabella di fig. 2.6.
I valori temporali forniti sono stati rilevati in determinate condizioni di test indicate dal costruttore come riferimento: di solito sono fornite delle relazioni, o dei grafici, per apportare le necessarie modifiche ai valori indicati quando le condizioni di impiego non sono uguali a quelle di test. In fig. 2.7 sono riportati due grafici relativi alla memoria 2102 (non sono gli unici) che indicano come varia il tempo di accesso t_A al variare della temperatura o della capacità del carico visto dalla linea D_{OUT}.

A.C. CONDITIONS OF TEST

Input Pulse Levels:		0.8 Volt to 2.0 Volt
Input Rise and Fall Times:		10nsec
Timing Measurement Reference Levels	Inputs:	1.5 Volts
	Output:	0.8 and 2.0 Volts
Output Load:		1 TTL Gate and C_L = 100 pF

Capacitance[2] T_A = 25°C, f = 1MHz

SYMBOL	TEST	LIMITS (pF) TYP.[1]	MAX.
C_{IN}	INPUT CAPACITANCE (ALL INPUT PINS) V_{IN} = 0V	3	5
C_{OUT}	OUTPUT CAPACITANCE V_{OUT} = 0V	7	10

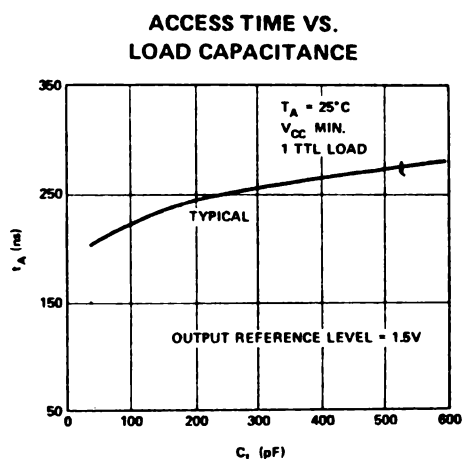
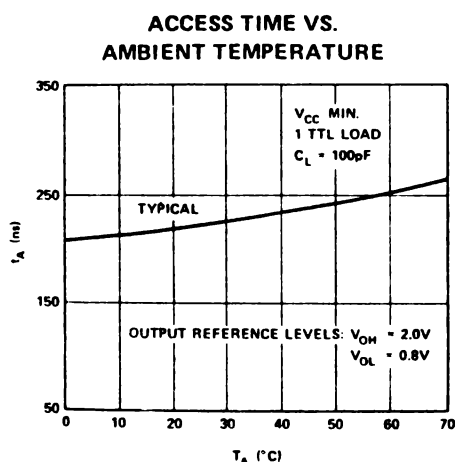



Fig. 2.7
Condizioni di test e variazioni del tempo di accesso t_A al variare della temperatura e della capacità di carico C_L nella memoria 2102. (Intel Component Data Catalog 1979).

2.3.3. *Funzionamento in standby*

Alcuni tipi di memorie a semiconduttori possono essere alimentati a potenza ridotta quando non viene eseguita alcuna operazione di lettura o di scrittura, senza che vadano perdute le informazioni immagazzinate (standby).

Ad esempio per portare la 2102-AL nello stato di standby si deve agire sul segnale CE/ e sulla tensione di alimentazione Vcc. In fig. 2.8 è riportato un diagramma che permette di dedurre come è possibile farla funzionare o meno in questo particolare stato a basso consumo di potenza.

Standby Characteristics 2102AL, 2102AL-2, and 2102AL-4 (Available only in the Plastic Package)

T_A = 0°C to 70°C

		2102AL, 2102AL-4 Limits			2102AL-2 Limits				
Symbol	Parameter	Min.	Typ.[1]	Max.	Min.	Typ.[1]	Max.	Unit	Test Conditions
V_{PD}	V_{CC} in Standby	1.5			1.5			V	
V_{CES}[2]	$\overline{CE}$ Bias in Standby	2.0			2.0			V	$2.0V \leq V_{PD} \leq V_{CC}$ Max.
		V_{PD}			V_{PD}			V	$1.5V \leq V_{PD} < 2.0V$
I_{PD1}	Standby Current		15	23		20	28	mA	All Inputs = V_{PD1} = 1.5V
I_{PD2}	Standby Current		20	30		25	38	mA	All Inputs = V_{PD2} = 2.0V
t_{CP}	Chip Deselect to Standby Time	0			0			ns	
t_R[3]	Standby Recovery Time	t_{RC}			t_{RC}			ns	

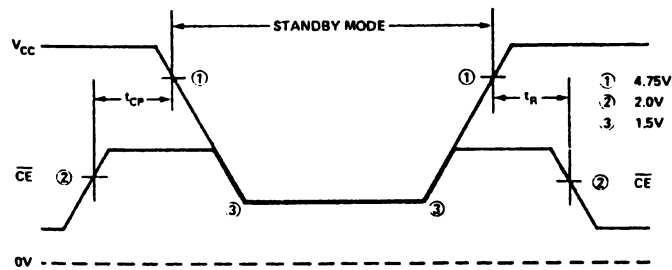
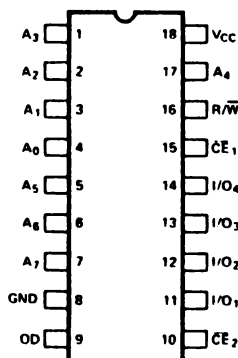
STANDBY WAVEFORMS

NOTES:
1. Typical values are for T_A = 25°C.
2. Consider the test conditions as shown: If the standby voltage (V_{PD}) is between 5.25V (V_{CC} Max.) and 2.0V, then $\overline{CE}$ must be held at 2.0V Min. (V_{IH}). If the standby voltage is less than 2.0V but greater than 1.5V (V_{PD} Min.), then $\overline{CE}$ and standby voltage must be at least the same value or, if they are different, $\overline{CE}$ must be the more positive of the two.
3. t_R = t_{RC} (READ CYCLE TIME).

Fig. 2.8
Caratteristiche di funzionamento standby della memoria 2102 (Intel Component Data Catalog 1979)

Come si può notare la 2102AL-2 per entrare nello stato di standby deve avere l'ingresso CE/ al valore logico alto; con un ritardo t_{CP}, che può essere anche nullo, la tensione Vcc dal valore nominale deve assumere un valore compreso fra 4,75 e 1,5 V per ridurre il consumo di potenza.
Mentre il passaggio dallo stato attivo a quello di standby può avvenire facendo assumere contemporaneamente all'ingresso di CE/ e a quello di alimentazione gli appropriati valori, nel passaggio inverso, una volta che Vcc abbia raggiunto il valore di 4.75 V, deve passare un tempo almeno pari ad un ciclo di lettura prima di poter effettuare l'attivazione della memoria.
E' ovvio che per utilizzare questo particolare modo di funzionamento sono necessari dei circuiti aggiuntivi per un preciso condizionamento delle varie tensioni.

2.4. Organizzazione di un banco di memoria

E' noto che un microprocessore funziona con un certo grado di parallelismo nel trasferimento ed elaborazione delle informazioni: molto spesso tale parallelismo è di otto bit, anche se si stanno attualmente diffondendo i microprocessori a 16 bit, mentre altri valori sono relativi solo a situazioni molto particolari e di limitata diffusione (ad esempio elaboratori a 12 bit). Si vuole analizzare la realizzazione di un banco di memoria per un processore ad 8 bit; i concetti esposti sono facilmente estensibili ad altri processori.

2111A/8111A-4
256 x 4 BIT STATIC RAM

A_0-A_7	ADDRESS INPUTS
OD	OUTPUT DISABLE
R/$\overline{W}$	READ/WRITE INPUT
$\overline{CE}_1$	CHIP ENABLE 1
CE_2	CHIP ENABLE 2
I/O_1-I/O_4	DATA INPUT/OUTPUT

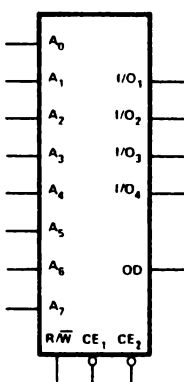
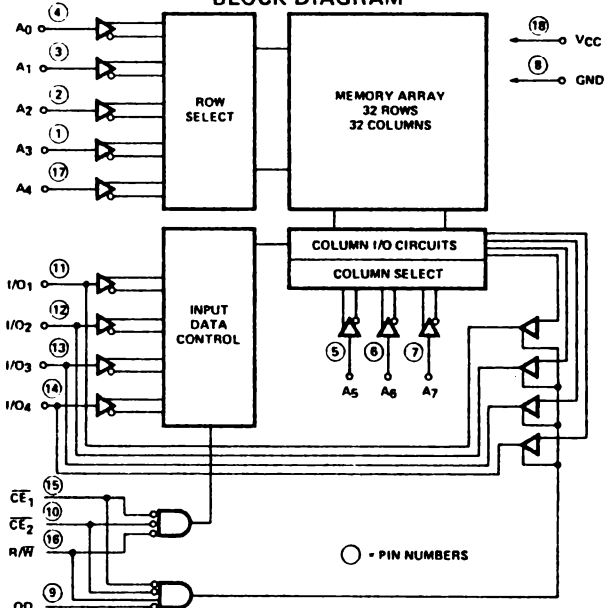



Fig. 2.9
Schema a blocchi della memoria RAM statica 2111 (256 x 4 bit) (Intel Component Data Catalog 1979)

Una prima esigenza da soddisfare è che la memoria deve essere organizzata in modo tale che quando ci si riferisce ad una sua posizione automatica-

mente si deve accedere ad un insieme di 8 bit, cioè ad un byte.
Sono attualmente disponibili dei circuiti integrati di memoria già organizzati in modo che la singola locazione indirizzabile sia un byte: in tal caso il problema posto è automaticamente risolto scegliendo questi componenti, che stanno sempre più diffondendosi per la loro semplicità di utilizzazione.
Ci sono però parecchi casi in cui i circuiti integrati di memoria non hanno una struttura a byte per cui si deve provvedere ad una opportuna disposizione circuitale esterna.
Tipi di memorie attualmente a disposizione sono organizzati o a quattro bit oppure ad un solo bit.
Al primo tipo appartiene ad esempio la memoria 2111 che presenta una capacità di 1024 bit organizzati in 256 parole di 4 bit ciascuno.
Lo schema a blocchi ed il significato dei vari pin sono riportati nella fig. 2.9.
Per ottenere una organizzazione a byte, si possono collegare due 2111 in parallelo come è riportato nella fig. 2.10.

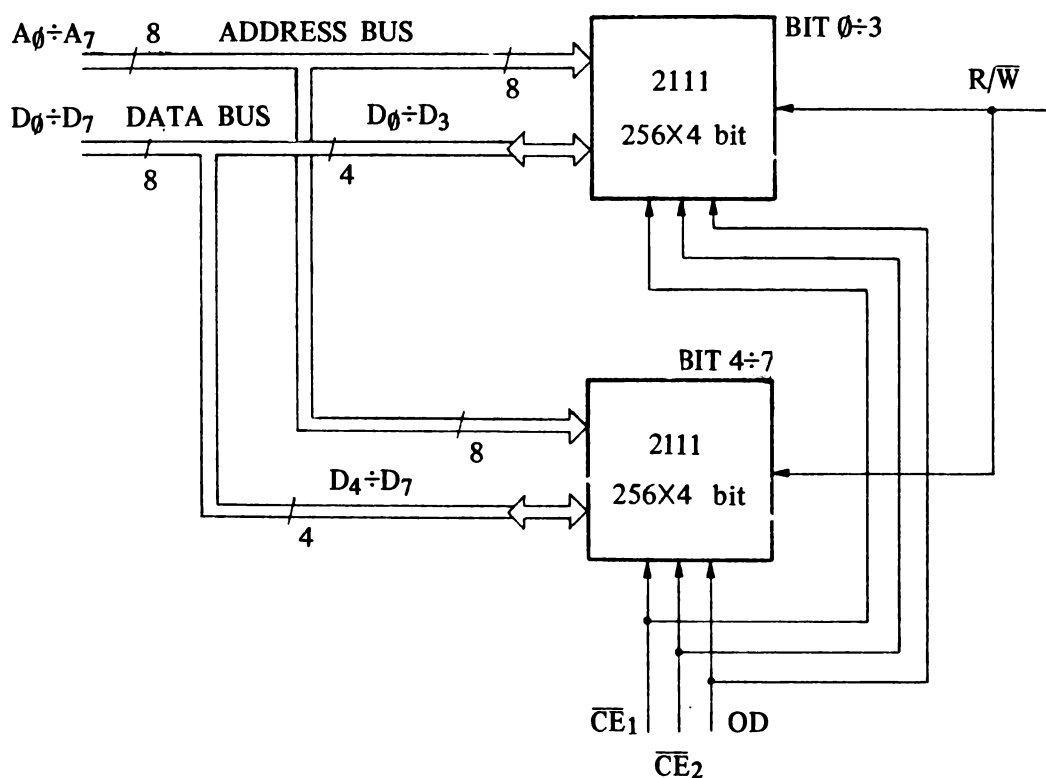



Fig. 2.10
Banco di memoria di 256 x 8 bit realizzato utilizzando memorie 2111 da 256 x 4 bit.

Con lo stesso indirizzo si individuano due distinti gruppi di 4 bit, uno per ciascun componente, che nel loro insieme costituiscono il byte, su cui saranno eseguite le operazioni desiderate a seconda dei valori assunti dai segnali di controllo, uguali per entrambe le memorie.
Si noti che il carico visto dai generatori di segnali per il bus degli indirizzi e di controllo è raddoppiato rispetto al caso di una memoria già organizzata a byte con uguali caratteristiche di ingresso ed uscita; ovviamente di ciò deve essere tenuto conto nel scegliere i vari driver dei bus.
Nel caso si utilizzino memorie organizzate a bit, in genere poco costose, si devono porre in parallelo ben 8 circuiti integrati per ottenere la richiesta organizzazione a byte.
Per ottenere l'organizzazione a byte con le 2102 si può utilizzare lo schema riportato nella fig. 2.11.

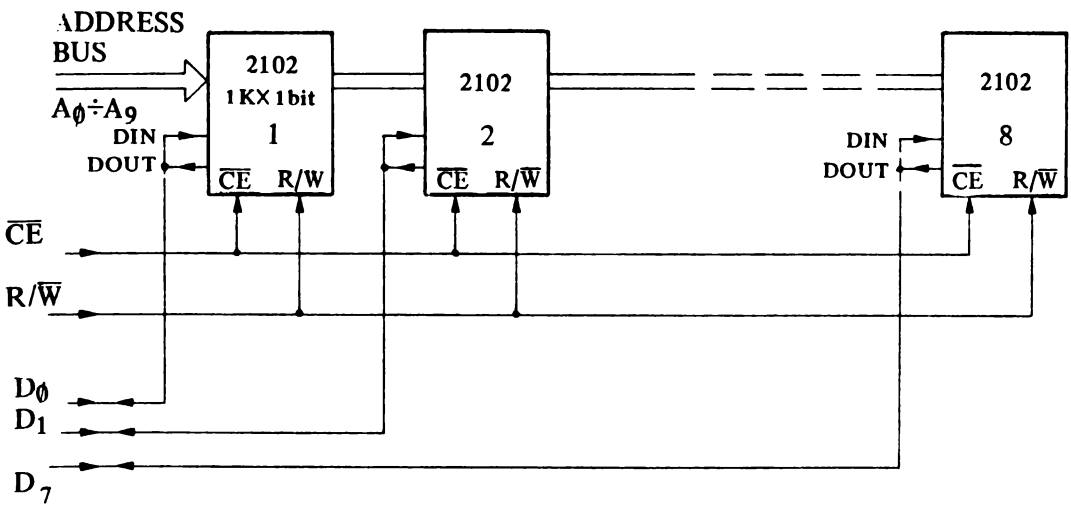



Fig. 2.11
Banco di memoria di 1 Kbyte realizzato con 8 memorie 1 K x 1 bit.

In questo caso con lo stesso indirizzo, di 10 bit, si individua la stessa locazione in otto distinte 2102, le cui uscite, nel loro insieme, forniscono il byte cercato.
Nel primo esempio riportato si è ottenuta una memoria di 256 byte men-

tre nel secondo di 1024 byte: volendo avere a disposizione un numero di locazioni maggiori si può ricorrere all'impiego di altri tipi di memorie, (ce ne sono alcune con addirittura 4 od 8 Kbyte), ma di solito è preferibile procedere ad una opportuna organizzazione di memorie a capacità ridotta per il loro costo minore.
In genere in un microelaboratore sono presenti sia la memoria programma che quella dei dati, cioè almeno una ROM ed una RAM. Sorge allora la necessità di utilizzare in modo appropriato i vari segnali presenti sui bus di indirizzo e di controllo in modo da attivare, quando occorre, solo quei circuiti integrati di memoria che sono interessati ad un trasferimento di dati: a ciò sono dedicati i circuiti di decodifica degli indirizzi.

2.5. Decodifica incompleta degli indirizzi

La decodifica degli indirizzi è uno dei problemi da affrontare quando si realizza un banco di memoria, sia esso composto di RAM o ROM; la soluzione al problema deve permettere di individuare l'unica posizione di memoria che corrisponde all'indirizzo presente sul bus omonimo.
Per analizzare le diverse tecniche utilizzate per la decodifica degli indirizzi si supponga di voler realizzare un banco di memoria composto da 6K byte di memoria ROM e di 512 byte di memoria RAM; per semplicità si supponga inoltre che tale memoria occupi posizioni contigue, a partire dalla posizione 0000H.*
In un microprocessore ad 8 bit sono disponibili di solito 16 bit per gli indirizzi, mediante i quali si possono individuare fino a 65.536 diverse posizioni di memoria (64 K).
Si tratta allora di utilizzare il valore presente sul bus degli indirizzi per selezionare il circuito integrato in cui è contenuto il byte su cui si desidera operare e, nell'ambito di tutte le celle contenute nel circuito integrato stesso, individuare proprio quella associata a tale valore.
Si supponga di utilizzare come RAM le memorie 2111, il cui schema a blocchi è stato riportato in fig. 2.9, mentre come memorie ROM le 8316, organizzate in byte, di capacità pari a 2 K, e con tre ingressi di selezione come indicato nello schema a blocchi di fig. 2.12.
Sono necessarie tre 8316 per la realizzazione dei 6 K byte di memoria ROM richiesti, mentre, poichè le memorie 2111 contengono ciascuna 256 x 4 bit, sono necessarie 4 di esse per ottenere i 512 byte di memoria RAM, ponendole in parallelo a gruppi di due.

---

* Le posizioni di memoria sono indicate in codice esadecimale.

PIN NAMES

A_0 A_{10}	ADDRESS INPUTS
O_1- O_8	DATA OUTPUTS
CS_1- CS_3	PROGRAMMABLE CHIP SELECT INPUTS

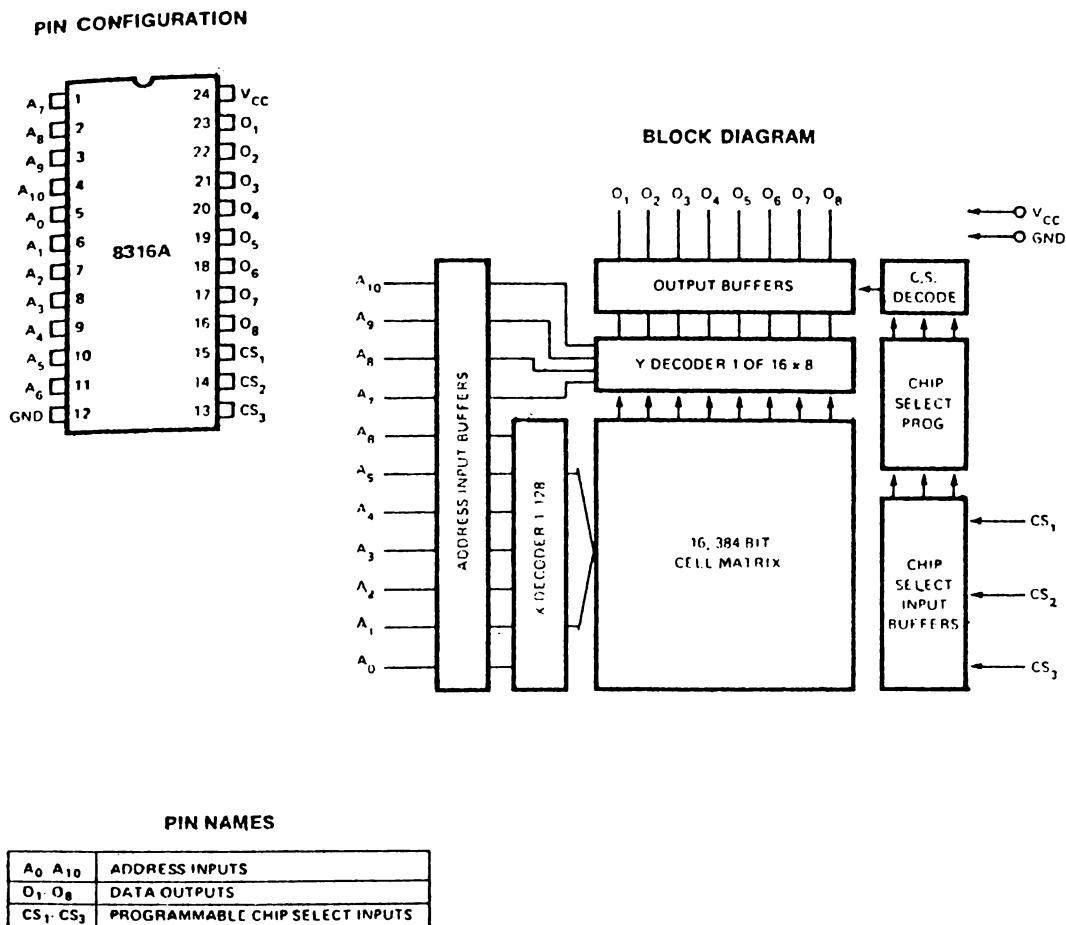



Fig. 2.12
Schema a blocchi della memoria ROM 8316.
(Intel, Microcomputer User's Manual 1976).

In fig. 2.13 è rappresentata l'organizzazione degli indirizzi della memoria dove è riportato, per ogni circuito integrato che la costituisce, l'indirizzo iniziale e finale sia in notazione esadecimale che binaria.
Per individuare una locazione all'interno di una qualsiasi delle tre 8316 sono necessari 11 bit (2^{11} = 2048): a tale scopo si utilizzano i segnali A0-A10 del bus degli indirizzi. Si tratta ora di adottare una opportuna organizzazione in modo che il primo byte della prima 8316 corrisponda alla posizione di memoria di indirizzo 0000H, il primo byte della seconda 8316 corrisponda alla posizione di memoria di indirizzo 0800H ed infine il primo byte della terza 8316 abbia l'indirizzo 1000H.

Fig. 2.13
Indirizzi associati ad ogni componente il banco di memoria.

In ogni caso, come si può notare dalla fig. 2.13, per gli indirizzi suddetti i primi 11 bit sono uguali per le tre 8316, mentre differiscono il dodicesimo e tredicesimo bit (A11 e A12). Si può sfruttare proprio questa differenza per selezionare, a seconda dei valori assunti da A11 e A12 quella tra le tre 8316, che contiene la cella individuata dai valori di A0 ÷ A10.
Le 8316 hanno tre ingressi di selezione, CS1, CS2, CS3 e si può richiedere al costruttore che essi siano attivi a livello logico alto oppure basso. E' possibile allora usare solo tali ingressi per ottenere la selezione desiderata, senza dover ricorrere a dell'hardware aggiuntivo: uno dei possibili modi di selezione è indicato in fig. 2.14 dove si è supposto che CS1 e CS2 siano attivi a livello logico basso, mentre CS3 a quello alto.

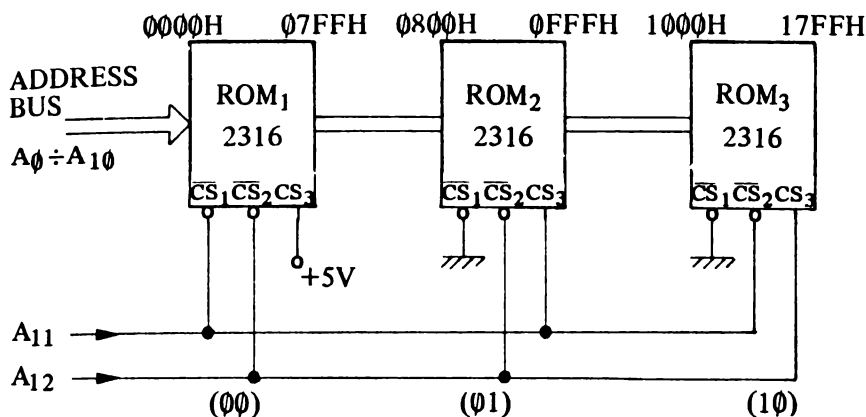



Fig. 2.14
Selezione delle prime 6 K - locazioni.

Quando le linee di indirizzo A11 e A12 si trovano entrambe a valore logico zero è selezionata la prima ROM; con A11 = 1 e A12 = 0 la seconda, mentre quando A11 = 0 e A12 = 1 è selezionata la terza.
Poichè sono sufficienti solamente due bit per la selezione, il terzo ingresso di CS presente va posto ad una tensione costante e cioè a 5V per CS3 del primo componente e a 0 V per CS1/ del secondo e del terzo.
Rimane ora da organizzare la selezione dei 512 byte di memoria RAM.
Analizzando la tabella di fig. 2.13 si può notare che per la individuazione di un byte all'interno del campo di indirizzi assegnati alla memoria RAM è necessario, oltre ai primi 8 bit, A0 ÷ A7, prendere in esame sia i valori assunti dai due bit di indirizzo A11 e A12, come nel caso precedente, sia quello del bit A8 al fine di individuare quale delle due coppie di 2111 deve entrare in attività.
Poiché in esse sono presenti solo due ingressi di selezione, CS1/ e CS2/, e questa è attuata dai valori assunti dai tre bit suddetti, è necessario realizzare degli opportuni circuiti di selezione.
Uno degli schemi che possono essere utilizzati è riportato nella fig. 2-15, mentre in fig. 2.16 è riportato lo schema completo del banco di memoria.

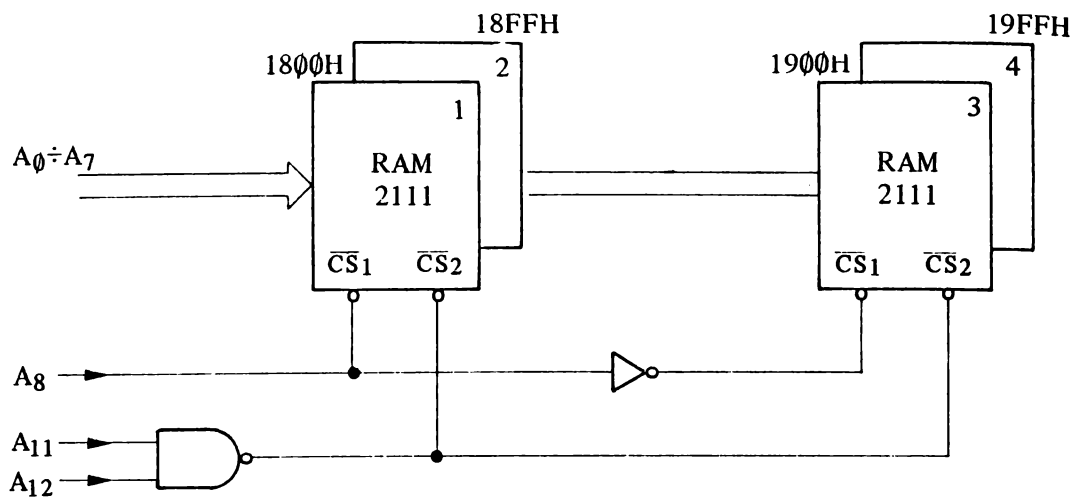



Fig. 2.15
Selezione delle 512 locazioni di RAM.

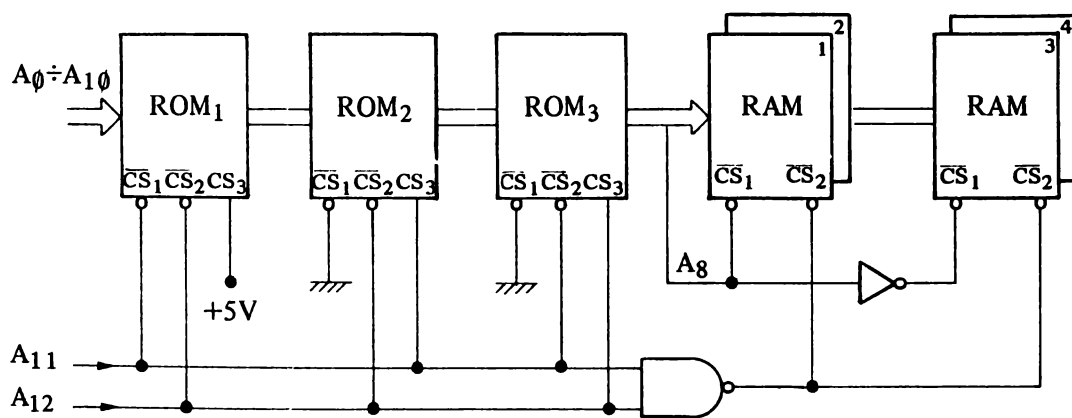



Fig. 2.16
Schema finale del banco di memoria.

Il tipo di decodifica adottato prende il nome di decodifica incompleta: infatti poichè i bit del bus degli indirizzi sono in totale 16 mentre nel caso attuale ne sono stati presi in considerazione solamente i primi 13, la cella di memoria, individuata dai valori assunti da A_0-A_{12}, è selezionata qualsiasi sia il valore di A_{13}, A_{14}, ed A_{15} ; quindi il banco di memoria è selezionato anche per gli indirizzi riportati nella fig. 2.17.

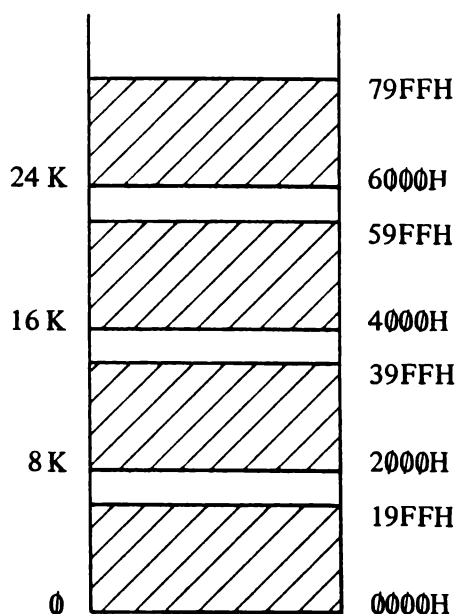



Fig. 2.17
Valori di indirizzo che selezionano il banco di memoria di 6,5 K.

Si ha una periodicità di 8 Kbyte all'interno dei quali sono indirizzabili i primi 6,5 K con la struttura hardware proposta.

2.5.1. *Decodifica completa*

La soluzione appena descritta ha il notevole vantaggio della semplicità, ma non consente di stabilire una corrispondenza biunivoca fra gli indirizzi e le posizioni di memoria.
Per ottenere tale biunivocità occorre prendere in considerazione anche i bit A_{13}, A_{14}, ed A_{15}: ciò comporta l'impiego di una rete combinatoria per l'analisi dei valori assunti dai tre bit significativi e quindi un maggior costo rispetto al caso precedente; una delle possibili modifiche da apportare a quanto prima visto è indicata in fig. 2.18.

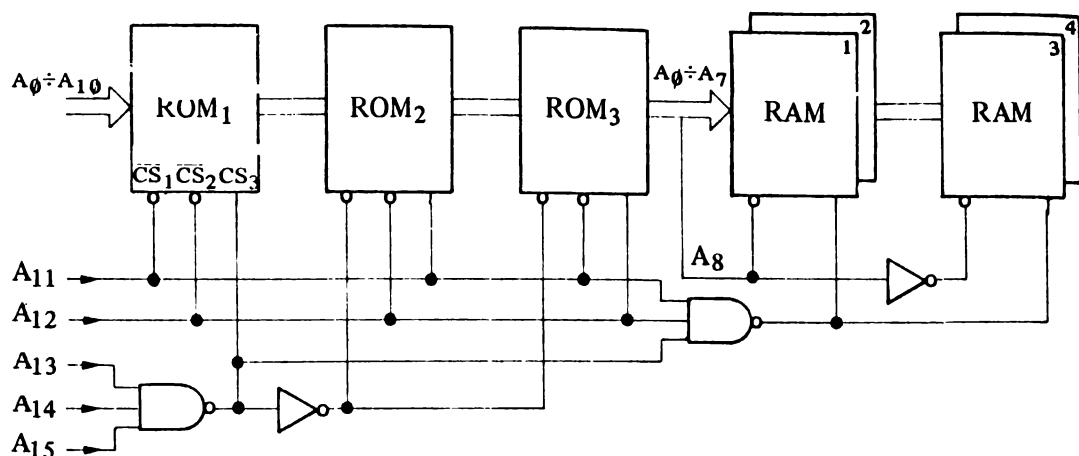



Fig. 2.18
Decodifica completa per il banco da 6,5 K.

Il circuito di tale figura è ora attivato solamente per indirizzi compresi fra 0000H e 19FFH, come era stato inizialmente richiesto. Questo metodo prende il nome di decodifica completa.
Come si è visto il circuito di selezione può essere dedotto sfruttando le caratteristiche delle memorie impiegate e per la sua realizzazione si utilizzano delle reti combinatorie.
Nel caso attuale si è sfruttato il fatto che i dispositivi di memoria utilizzati presentavano due o tre ingressi di selezione.
In molti casi però il circuito integrato di memoria presenta un solo ingresso per l'abilitazione: senza riferirci a casi particolari si supponga ora che questo, CS/, sia attivo a livello logico basso e si voglia realizzare ancora un banco di memoria, che occupi le stesse posizioni finora prese in considerazione.
Si deve allora progettare una rete combinatoria che dai segnali di indirizzo fornisca i richiesti segnali di abilitazione dei vari componenti: una delle possibili realizzazioni è riportata nella fig. 2.19.

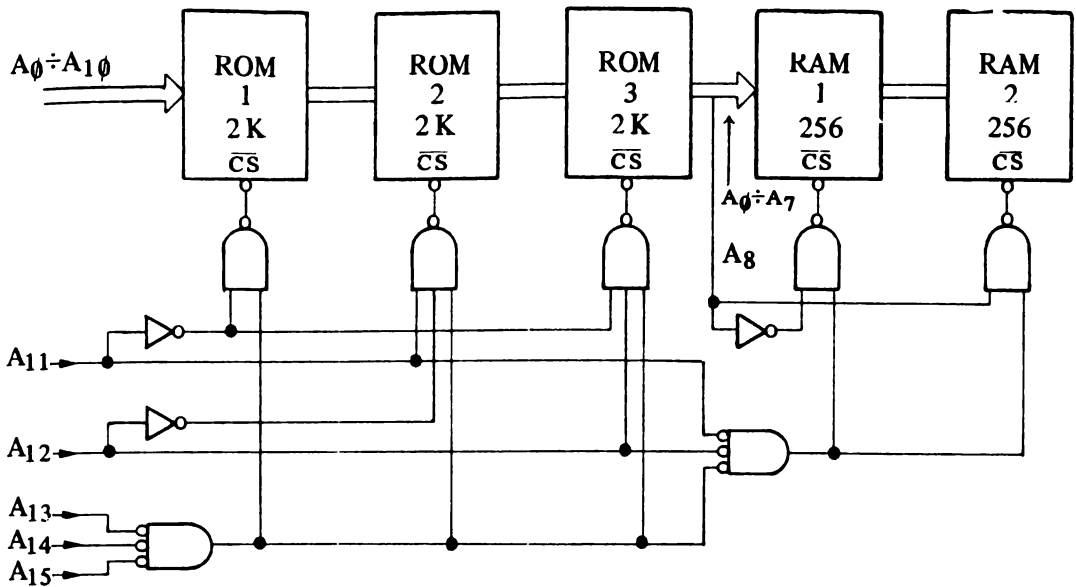



Fig. 2.19
Decodifica degli indirizzi di un banco di memoria da 6,5 K nel caso i dispositivi presentino un solo ingresso di CS/.

2.5.2. *Uso dei decodificatori*

Per la selezione delle memorie si utilizzano molto spesso dei circuiti integrati, detti decodificatori, che permettono di semplificare notevolmente l'hardware necessario.
Un tipico esempio di decodificatore utilizzabile allo scopo è l'8205, il cui simbolo logico e la relativa tabella di verità sono riportati nella fig. 2.20. Tale dispositivo presenta tre ingressi di indirizzo, A_0, A_1, A_2 i quali, a seconda del valore cui sono portati, individuano ordinatamente l'unica delle otto uscite, $0_0 \div 0_7$, la quale si porterà al valore logico basso se contemporaneamente sono al valore logico che loro compete i tre ingressi di abilitazione $E_1/$, $E_2/$, E_3 (rispettivamente 0, 0, 1).

8205

HIGH SPEED 1 OUT OF 8 BINARY DECODER

ADDRESS			ENABLE			OUTPUTS							
A_0	A_1	A_2	E_1	E_2	E_3	0	1	2	3	4	5	6	7
L	L	L	L	L	H	L	H	H	H	H	H	H	H
H	L	L	L	L	H	H	L	H	H	H	H	H	H
L	H	L	L	L	H	H	H	L	H	H	H	H	H
H	H	L	L	L	H	H	H	H	L	H	H	H	H
L	L	H	L	L	H	H	H	H	H	L	H	H	H
H	L	H	L	L	H	H	H	H	H	H	L	H	H
L	H	H	L	L	H	H	H	H	H	H	H	L	H
H	H	H	L	L	H	H	H	H	H	H	H	H	L
X	X	X	L	L	L	H	H	H	H	H	H	H	H
X	X	X	H	L	L	H	H	H	H	H	H	H	H
X	X	X	L	H	L	H	H	H	H	H	H	H	H
X	X	X	H	H	L	H	H	H	H	H	H	H	H
X	X	X	H	L	H	H	H	H	H	H	H	H	H
X	X	X	L	H	H	H	H	H	H	H	H	H	H
X	X	X	H	H	H	H	H	H	H	H	H	H	H

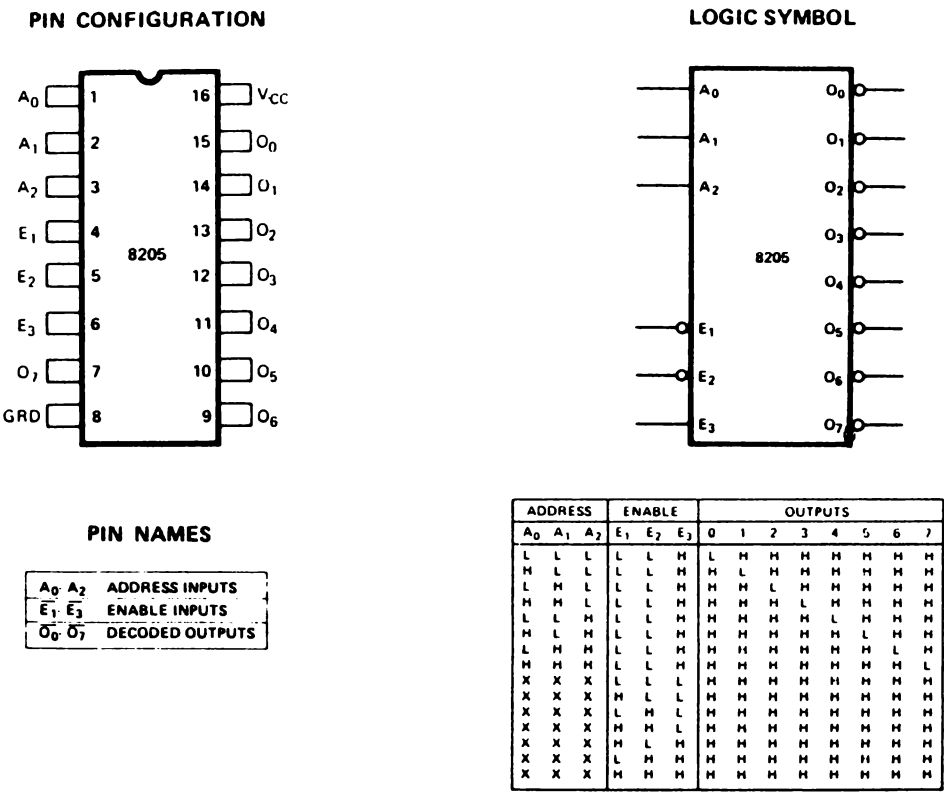



Fig. 2.20
Decodificatore 8205 (Intel Component Data Catalog 1979).

La precedente rete combinatoria può essere sostituita dal solo decodificatore 8205, come è indicato nella fig. 2.21, semplificando la rete di selezione della memoria.

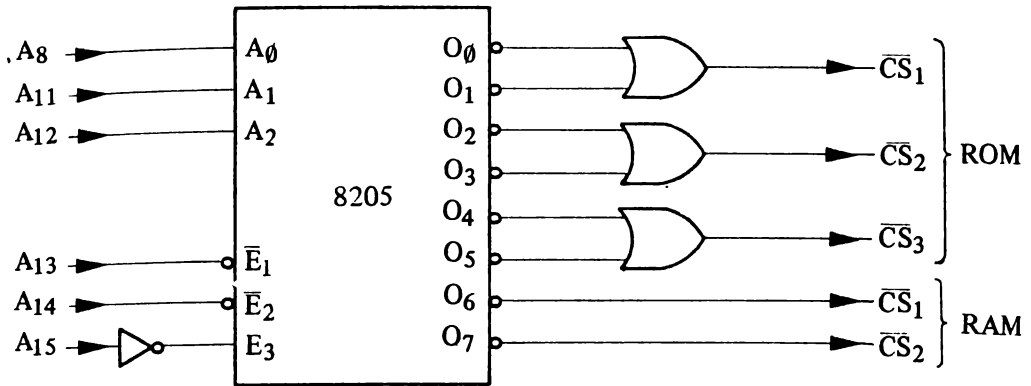



Fig. 2.21
Utilizzazione del decodificatore 8205 per la selezione del banco di memoria.

2.5.3. *Decodifica lineare*

Nel caso si debbano realizzare banchi di memoria di limitate capacità è possibile adottare la tecnica della decodifica lineare: essa consiste nell'ottenere la selezione di un circuito di memoria utilizzando direttamente un bit del bus degli indirizzi, e ciò per tutti i circuiti presenti.
Si voglia ad esempio realizzare un banco di memoria, di 8 Kbyte, utilizzando 4 componenti da 2 Kbyte ciascuno; un possibile schema di selezione potrebbe essere quello riportato nella fig. 2.22.

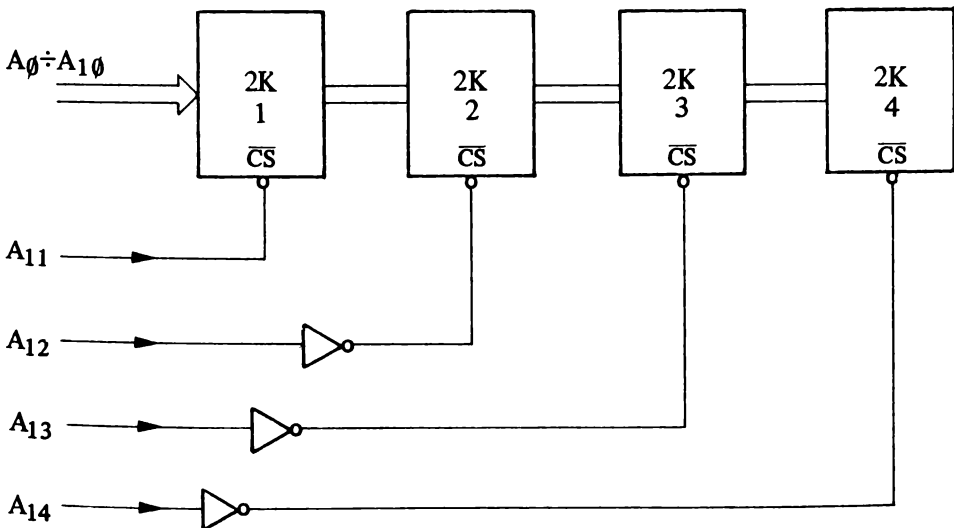



Fig. 2.22
Selezione lineare.

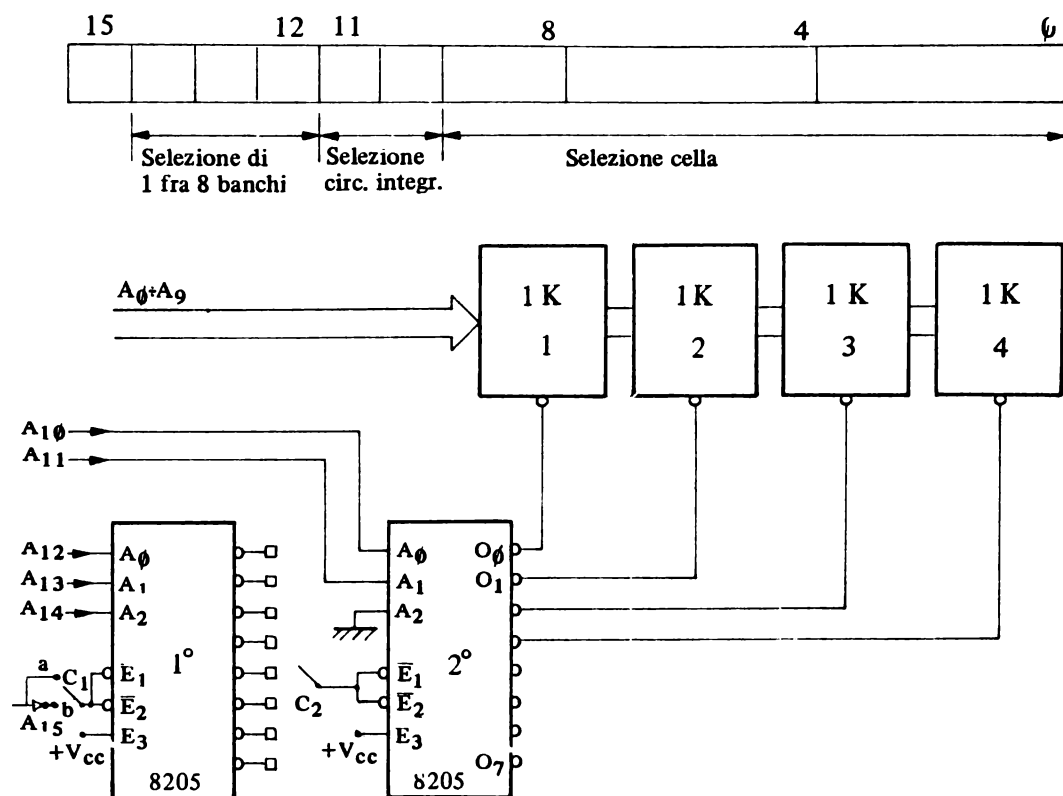



Fig. 2.23
Decodifica ad indirizzo base variabile.

Con il primo decodificatore si individuano le varie posizioni nei primi 32K mentre il secondo permette l'allocazione nei 32K superiori.
Con un solo commutatore a 16 posizioni si può ora allocare il banco di 4K nell'intero campo di 64K.
Un altro modo di procedere, piuttosto conveniente, utilizza un comparatore; questo è un circuito integrato in grado di effettuare il confronto fra due grandezze binarie di ingresso e di indicare in uscita se la prima è maggiore, minore oppure uguale alla seconda.

Alla semplicità offerta da tale soluzione si contrappone però l'inconveniente dovuto al fatto che le quattro aree di memoria non sono fra loro contigue e questo può creare qualche difficoltà da un punto di vista software. Inoltre le aree di memoria hanno campi di indirizzo prefissati: infatti la prima inizia dall'indirizzo 0000H e le altre da 1800H, 2800H e 4800H rispettivamente.
Anche il numero di componenti che si possono selezionare è limitato: nel caso attuale si possono al massimo usare 5 memorie da 2 Kbyte.

2.6. Decodifica ad indirizzo base variabile

Nella realizzazione di una scheda che contiene un banco di memoria risulta conveniente far si che il campo di indirizzi cui esso è associato possa essere facilmente cambiato senza dover modificare il circuito di decodifica. In questo modo è possibile ampliare con estrema facilità la quantità di memoria presente nel sistema, utilizzando più schede uguali ed imponendo loro un indirizzo di base diverso, essendo questo l'indirizzo associato alla prima locazione di memoria di ogni banco.

Si voglia ad esempio far si che un banco di memoria della capacità di 4 Kbyte sia in grado di assumere un indirizzo base, variabile con incrementi di 4 Kbyte, da 0 fino a 60 K; possa cioè assumere gli indirizzi base 0000H, 1000H, 2000H, ...fino a F000H.
Supponendo che il banco sia realizzato con circuiti di memoria della capacità di 1 Kbyte, un possibile schema è indicato nella fig. 2.23.
Esso utilizza due decodificatori, ad esempio del tipo 8205, in cascata e due commutatori. Il primo decodificatore impone l'indirizzo base nell'ambito dei primi o degli ultimi 32 K.
Ognuna delle uscite del primo decodificatore sarà attiva a livello logico basso in corrispondenza ad un prefissato valore dei segnali A_{14} A_{13} e A_{12}, coprendo in tal modo 8 x 4 Kbyte = 32 Kbyte di memoria.
Per avere la possibilità di allocare il banco in qualsiasi posizione nell'ambito di 64 K è utilizzato il bit A_{15}: mediante esso si stabilisce infatti in quale dei due blocchi da 32 K deve trovarsi il banco in oggetto. Con il commutatore C_1 in posizione "a" si alloca il banco nei primi 32 Kbyte, mentre se C_1 è in posizione "b" il banco è allocato negli ultimi 32 Kbyte.
Si noti che è necessario agire su due commutatori per poter predisporre l'indirizzo base del banco nell'intero campo di 64 K.
Se si vuole agire su un solo commutatore si può utilizzare lo schema di fig. 2.24, dove si sono collegati in modo opportuno gli ingressi di abilitazione del primo e del secondo decodificatore al segnale A_{15}.

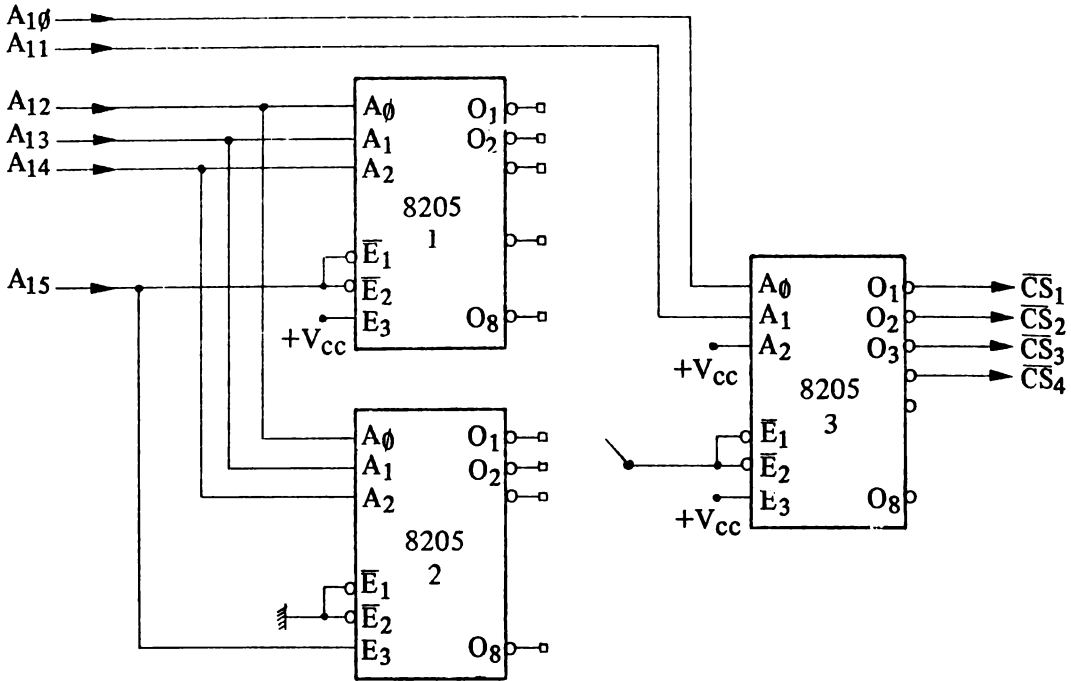



Fig. 2.24
Decodifica ad indirizzo base variabile con un solo commutatore.

Lo schema di tale componente e la corrispondente tabella di verità che dà conto del suo funzionamento sono riportati in fig. 2.25.
Le due grandezze di ingresso A e B su cui è effettuato il confronto sono entrambe da quattro bit; esistono tre uscite una sola delle quali può in ogni istante essere attiva per fornire il risultato del confronto. Sono presenti altri tre ingressi che servono per il collegamento in cascata di più comparatori.
Si supponga di voler realizzare il solito banco di memoria di 4 Kbyte, utilizzando quattro memorie ognuna con capacità di 1 Kbyte. Un possibile circuito è quello indicato nella fig. 2.26.

Ingressi di confronto				Collegamento in cascata			Uscita		
A_3, B_3	A_2, B_2	A_1, B_1	A_0, B_0	$A>B$	$A<B$	$A=B$	$A>B$	$A<B$	$A=B$
$A_3>B_3$	X	X	X	X	X	X	H	L	L
$A_3<B_3$	X	X	X	X	X	X	L	H	L
$A_3=B_3$	$A_2>B_2$	X	X	X	X	X	H	L	L
$A_3=B_3$	$A_2<B_2$	X	X	X	X	X	L	H	L
$A_3=B_3$	$A_2=B_2$	$A_1>B_1$	X	X	X	X	H	L	L
$A_3=B_3$	$A_2=B_2$	$A_1<B_1$	X	X	X	X	L	H	L
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0>B_0$	X	X	X	H	L	L
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0<B_0$	X	X	X	L	H	L
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	H	L	L	H	L	L
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	L	H	L	L	H	L
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	L	L	H	L	L	H

Fig. 2.25
Comparatore e sua tabella di funzionamento.

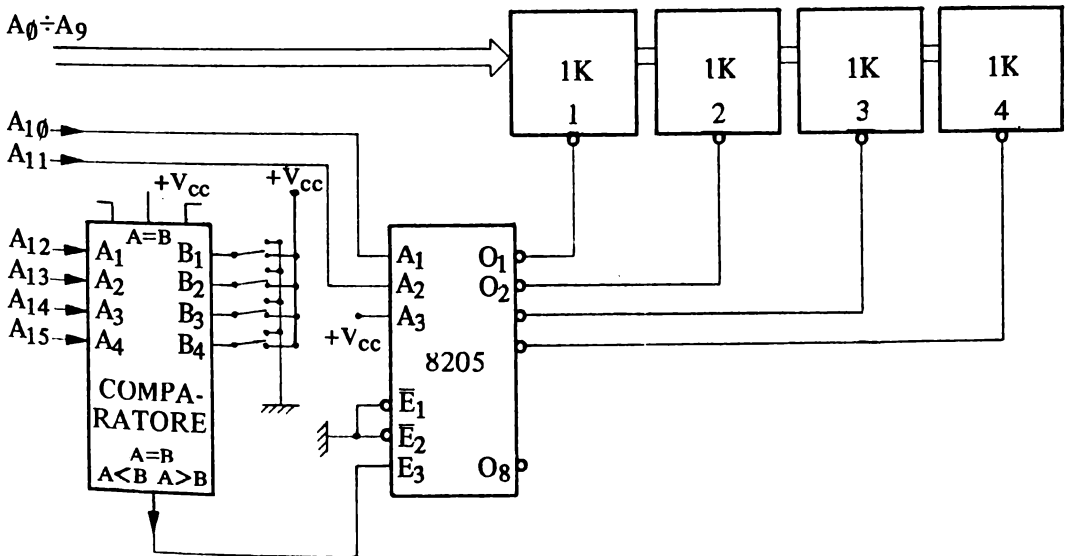



Fig. 2.26
Decodifica ad indirizzo base variabile mediante l'uso di un comparatore.

La selezione dei quattro circuiti integrati di memoria avviene per il tramite di un decodificatore 8205 all'ingresso del quale si hanno i due bit di indirizzo A_{10} e A_{11}. Le uscite del decodificatore diventano attive solamente se l'ingresso di abilitazione E_3 assume il valore logico alto: ciò avviene solo se c'è identità nei valori presenti agli ingressi A e B del comparatore. Gli ingressi A sono collegati ai quattro bit più significativi del bus degli indirizzi; a quelli B è invece imposta la configurazione scelta mediante la posizione di quattro deviatori.

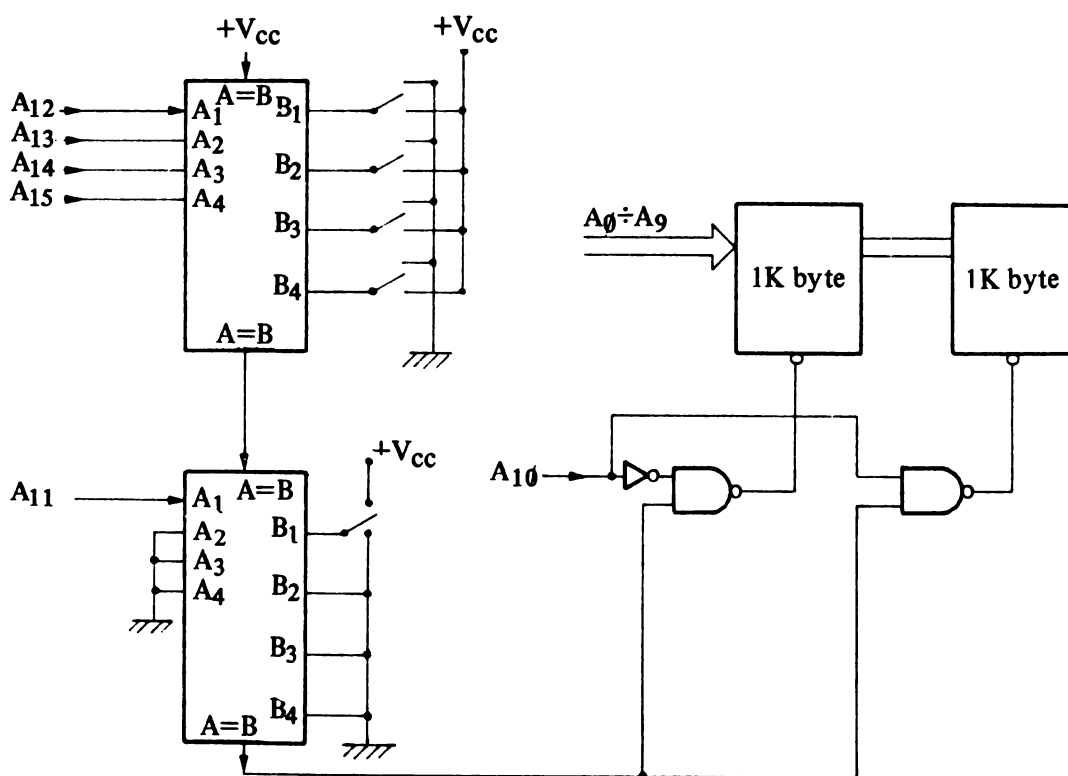



Fig. 2.27
Circuito di decodifica tramite comparatori.

A seconda della posizione di questi ultimi, e quindi della grandezza di ingresso B, si ha una diversa allocazione del banco di memoria da 4 Kbyte.
Il comparatore presenta il vantaggio che con semplici cambiamenti delle connessioni realizzate dai quattro deviatori è possibile allocare il banco di memoria nelle sedici posizioni richieste: 0000H, 2000H, 4000H, ...F000H.
Con lo schema proposto in fig. 2.26 si può ottenere lo spostamento dell'indirizzo base a gradini di 4 K; è possibile, impiegando più comparatori, ottenere una definizione maggiore: se, ad esempio si vuole realizzare un banco di memoria di 2 K byte, utilizzando 2 memorie da 1 K, allocabile in qualsiasi posizione entro i 64 K a partire da indirizzi base multipli di 2 K, si può utilizzare lo schema di fig. 2.27.
In questo caso con i cinque deviatori presenti si può ottenere l'allocazione che si desidera a partire da posizioni multiple di 2 K. Si noti che si potrebbero usare altri schermi che utilizzano gli ingressi predisposti per il collegamento in cascata dei comparatori.

2.7. Decodifica con ROM

Per realizzare la decodifica degli indirizzi è anche possibile utilizzare una memoria ROM opportunamente programmata: in ogni locazione della ROM è memorizzata una particolare configurazione di bit in modo che sia attivata una sola delle sue uscite, utilizzate per abilitare i circuiti integrati che compongono il banco di memoria; in pratica si realizza mediante una ROM la funzione svolta da un decodificatore.
Il notevole vantaggio offerto da tale metodo consiste nel poter modificare facilmente le dimensioni di banchi di memoria senza dover aggiungere ulteriore hardware, come è necessario con l'uso dei decodificatori; lo svantaggio è che tale soluzione non si presta per la realizzazione di piccole serie.
Per illustrare le modalità d'uso di una memoria ROM come decodificatore si consideri il caso di voler realizzare un banco di memoria di 4 Kbyte con quattro circuiti integrati ciascuno con capacità di 1 Kbyte; si voglia ad esempio allocare tale banco nell'area da 2000H a 2FFFH. Evidentemente la ROM dovrà contenere parole di almeno 4 bit, dato che quattro sono i segnali necessari per selezionare i diversi circuiti integrati. Se, come al solito, il bus degli indirizzi è formato da 16 bit e si vuole realizzare una decodifica completa, occorrono 6 ingressi di indirizzo alla ROM in quanto servono 10 bit per individuare una qualsiasi posizione all'interno di ogni circuito integrato di memoria. Per la decodifica richiesta è allora sufficiente una ROM di 64 parole da 4 bit ciascuna.
In fig. 2.28 è riportato uno schema dove sono indicati solamente i segnali che interessano la decodifica.

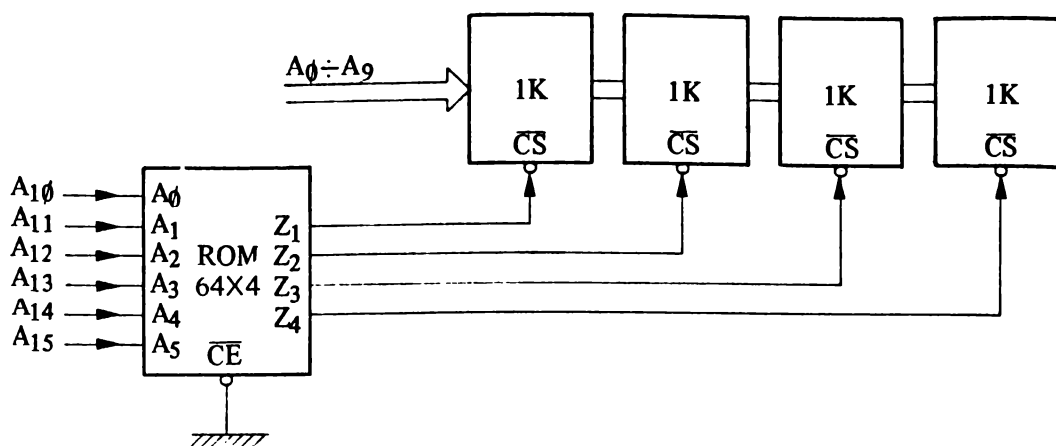



Fig. 2.28
Circuito di decodifica con ROM.

Si tratta ora di individuare le configurazioni di bit da programmare nella ROM: per fare questo si devono analizzare quali sono le configurazioni, di ingresso della stessa, per le quali debbono essere attivati i segnali di selezione, CS/, dei circuiti integrati di memoria; si supponga che, come normalmente avviene, l'attivazione di un ingresso di CS/ consista nel portarlo al livello logico basso. Gli indirizzi che interessano i vari circuiti integrati e le varie uscite sono riportati nella fig. 2.29.
Come si può osservare, quando all'ingresso della ROM è applicata la configurazione 001000, deve essere selezionato il primo circuito integrato. All'indirizzo 001000 della ROM deve quindi essere memorizzato il valore 1110 con il quale si attiva l'uscita Z_1, che è collegata proprio a CS1/; all'indirizzo 001001 dovrà essere memorizzato il valore 1101 per selezionare il secondo circuito integrato di memoria.
Negli indirizzi 001010 e 001011 le parole da memorizzare sono rispettivamente, 1011 e 0111. Nelle altre restanti 60 posizioni di memoria della ROM, aventi indirizzi diversi da quelli indicati nessuno dei quattro circuiti integrati deve essere selezionato per cui deve essere memorizzata la configurazione 1111.
Si noti la bassa utilizzazione della ROM: infatti solamente 4 su 64 posizioni di memoria sono utilizzate.
Per variare l'indirizzo base del banco di memoria si tratta solo di riprogrammare la ROM memorizzando le quattro configurazioni di selezione in

opportuni indirizzi dipendenti dall'indirizzo base desiderato.

Fig. 2.29
Tabella di verità della ROM di fig. 2.28.

Con questa tecnica è anche possibile espandere la capacità del banco di memoria, utilizzando dei circuiti integrati di capacità più elevata, senza dover ricorrere ad una completa ristrutturazione dell'hardware.
L'eventualità di dover effettuare questa espansione si presenta abbastanza di frequente nella realizzazione di microelaboratori: infatti nella progettazione di un microelaboratore è molto difficile valutare inizialmente, in modo abbastanza preciso, la capacità di memoria richiesta per una certa applicazione. E' comodo allora prevedere banchi di memoria che siano variabili come dimensione ricorrendo all'utilizzazione di circuiti integrati con capacità superiore e con configurazione dei piedini compatibile.
Si consideri ad esempio di voler espandere il precedente banco di memoria impiegando dei circuiti integrati di capacità doppia così da ottenere un

banco di 8 Kbyte. (Si tenga presente che un circuito integrato di memoria con capacità di 2 Kbyte ha sicuramente un piedino in più rispetto a quello di 1 Kbyte; di solito la piedinatura è compatibile per i segnali di uscita, di controllo e per quelli di indirizzo comuni).
Per raddoppiare il banco di memoria, se si fosse utilizzato un normale decodificatore, si sarebbero dovute aggiungere quattro porte AND per poter attivare i chip-select dei circuiti integrati ogni volta che su due uscite consecutive del decodificatore sia presente un segnale a valor logico 0 (fig. 2.30).

Fig. 2.30
Raddoppio del banco di memoria con l'uso di un normale decodificatore.

Nel caso di decodifica con ROM basta utilizzare un ingresso della stessa per stabilire il tipo di decodifica che essa deve svolgere. Una delle possibili realizzazioni è riportata in fig. 2.31.
La selezione del tipo di decodifica che la ROM deve realizzare in base al tipo di memoria impiegata è ottenuta mediante l'imposizione di un valore logico fisso all'ingresso A_5 della ROM: se ad esso è imposto il valore logico 0 si tratta di 4 Kbyte, se no di 8 Kbyte.
Si noti ora che A_{15} e il circuito associato ha la funzione di scegliere se il banco di memoria (sia esso di 4 che di 8 Kbyte) è allocato nella prima o nella seconda metà dell'area totale di memoria indirizzabile.
Nel caso si vogliano allocare i 4 K al solito nelle locazioni da 2000H a 2FFFH le posizioni di memoria interessate ed il loro contenuto sono quelle prima esaminate.

Fig. 2.31
Possibiiità di effettuare decodifiche diverse con una ROM.

Nel caso del banco di 8 K, se si vuole mantenere lo stesso indirizzo base, i segnali di indirizzo ed il corrispondente contenuto delle locazioni di memoria ROM sono riportati nella fig. 2.32.

	INGRESSI						USCITE			
	A_5	A_4	A_3	A_2	A_1	A_0	Z_4	Z_3	Z_2	Z_1
2000H	1	0	1	0	0	0	1	1	1	0
27FFH	1	0	1	0	0	1				
2800H	1	0	1	0	1	0	1	1	0	1
2FFFH	1	0	1	0	1	1				
3000H	1	0	1	1	0	0	1	0	1	1
37FFH	1	0	1	1	0	1				
3800H	1	0	1	1	1	0	0	1	1	1
3FFH	1	0	1	1	1	1				

Fig. 2.32
Tabella di verità della ROM di fig. 2.31.

Come si può notare, agli indirizzi indicati vanno memorizzate le configurazioni che abilitano i circuiti integrati interessati; inoltre in questo caso il valore di A_{10} non è necessario per identificare il circuito integrato che dovrà essere selezionato: infatti visto che si tratta di circuiti integrati con capacità pari a 2 Kbyte non è necessario A_{10} per individuare a quale gruppo di 2048 locazioni ci si riferisce.
E' allora evidente che per la decodifica sono in questo caso necessarie 8 parole di ROM per cui in totale sono utilizzate 12 parole su 64. Si noti che sia all'indirizzo 101000 che all'indirizzo 101001 nella ROM è memorizzata la stessa configurazione, 1110, che permette di selezionare il primo dei quattro circuiti integrati da 2 K; alle locazioni 101010 e 101011 è memorizzata la configurazione 1101 e così via.

2.8. Trasferimento dei dati fra la CPU e la memoria

Finora si sono analizzati i vari modi con cui è possibile generare i segnali di attivazione di un circuito integrato di memoria utilizzando le informazioni inviate dalla CPU attraverso il bus degli indirizzi. Si vuole ora esaminare come si attua il trasferimento dei dati fra i due dispositivi di cui sopra.
Gli attuali microprocessori acquisiscono e forniscono i dati attraverso gli stessi piedini, per cui necessariamente il bus dei dati è bidirezionale. Si supponga che si abbia il bus degli indirizzi separato da quello dei dati e di voler collegare una memoria ROM all'unità centrale. Uno schema utilizzabile è riportato nella fig. 2.33.

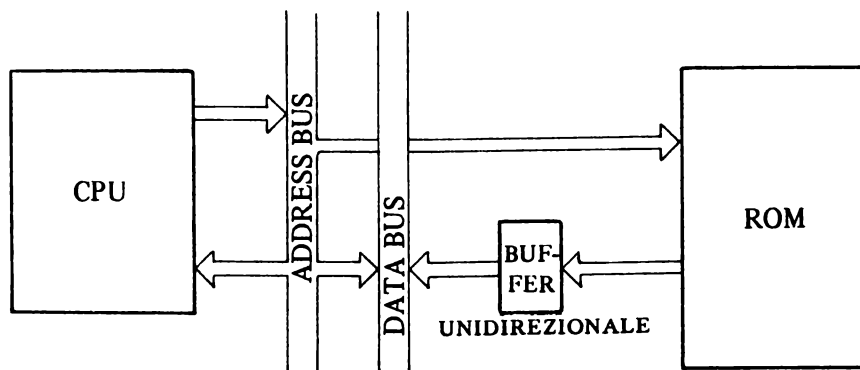



Fig. 2.33
Il collegamento di una memoria ROM al bus dei dati.

Il flusso dei dati è uniderezionale: è conveniente, come indicato in figura, interporre fra la memoria ed il bus del sistema un buffer al fine di proteggere le uscite della stessa durante la fase di scrittura in altre memorie (in fig. 2.33 non sono stati messi in evidenza, per semplificare, i segnali di controllo e l'utilizzazione degli indirizzi per la selezione).
Nel caso di una memoria RAM sono possibili due schemi (fig. 2.34).

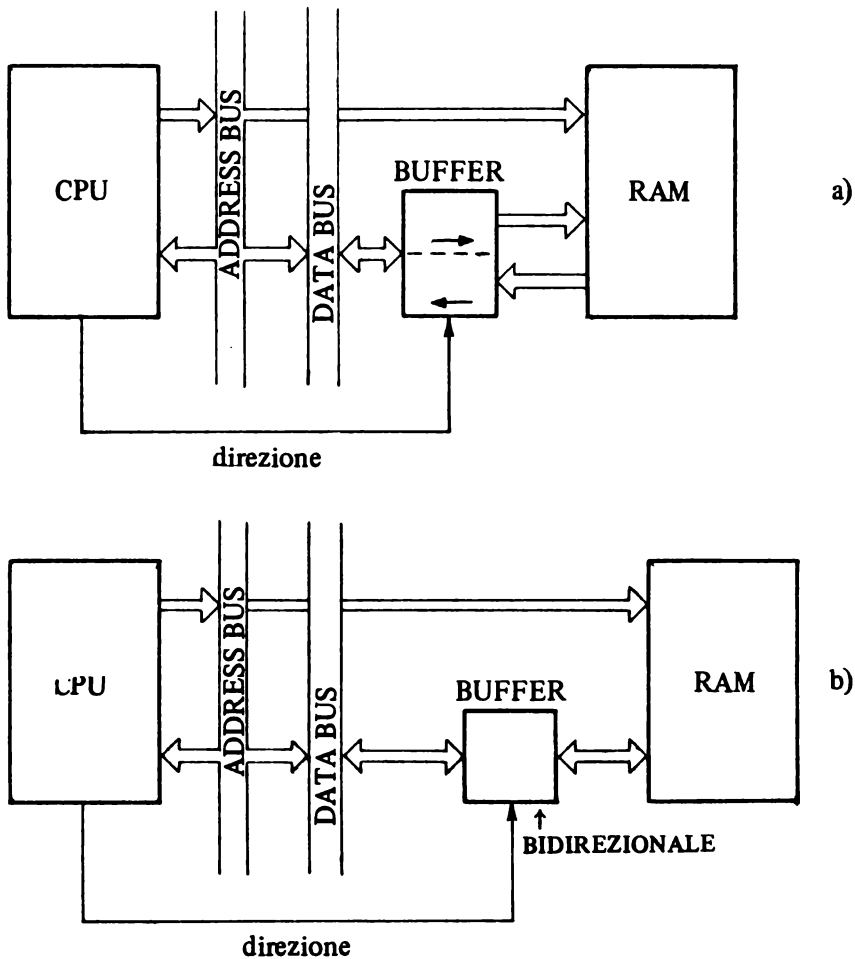



Fig. 2.34
Collegamenti di memorie RAM al bus dei dati.

Esistono infatti delle memorie RAM con ingressi e uscite dei dati separati, altre invece con gli stessi coincidenti; i buffer da utilizzare debbono quindi presentare delle strutture diverse. Inoltre è necessario un comando per stabilire la direzione del trasferimento attraverso il buffer. Tale comando a volte è fornito direttamente dalla CPU, a volte da una semplice rete combinatoria attivata dai segnali di controllo.
Certi tipi di microprocessori presentano il bus degli indirizzi multiplexato con quello dei dati. In tale caso lo schema utilizzabile è quello di fig. 2.35:

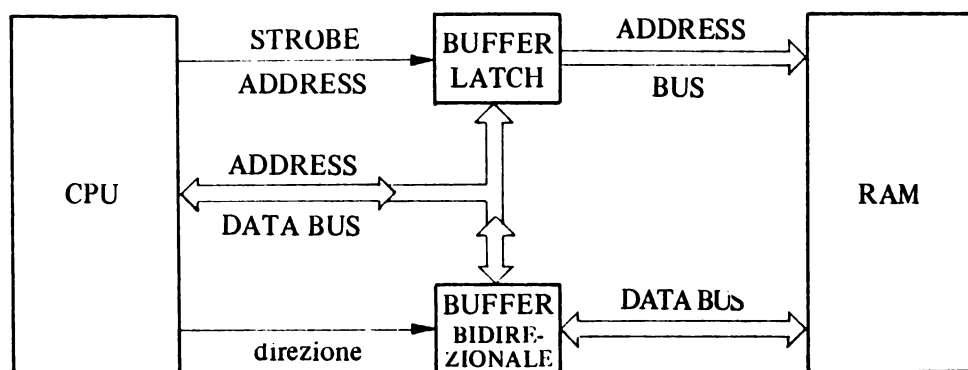



Fig. 2.35
Collegamento della memoria con microprocessori aventi i bus dei dati e degli indirizzi multiplexati.

dove il buffer-latch è necessario per separare i due tipi di informazioni che si presentano sulle medesime linee. Qualsiasi sia l'operazione di trasferimento da eseguire, prima è inviato sul bus l'indirizzo che individua la posizione di memoria interessata e successivamente avviene il trasferimento richiesto. Si noti che l'indirizzo deve rimanere a disposizione della memoria per un tempo in genere non coincidente con quello nel quale lo mette a disposizione la CPU: si deve allora provvedere a conservare tale informazione ed a ciò è utilizzato un buffer di tipo latch. L'istante in cui questo ultimo deve memorizzare l'indirizzo è indicato dalla segnalazione di "STROBE" proveniente dal master. Non è invece necessario che la caratteristica di latch sia presentata anche dal buffer utilizzato per il trasferimento dei dati. In questo caso è infatti necessario solo indicare la direzio-

ne del trasferimento se si tratta di una memoria RAM, mentre tale informazione non è richiesta per una memoria ROM.

2.9. Organizzazione di un banco di memoria a Word

Esistono dei processori, in particolare quelli a 16 bit, che hanno la possibilità di accedere alla memoria sia per prelevare dei byte che delle parole da 16 bit, a seconda del tipo di istruzione in esecuzione. In genere, nel caso si voglia accedere ad una parola, è richiesto che l'indirizzo cui ci si riferisce sia pari, mentre per l'accesso ad un byte l'indirizzo può essere qualsiasi. Un possibile schema che permette tale modo di funzionamento è quello di fig. 2.36.

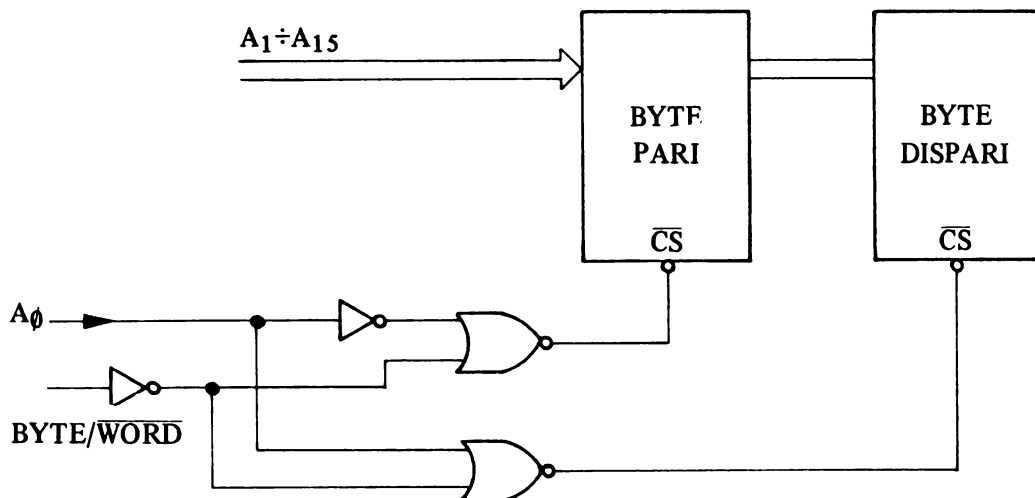



Fig. 2.36
Organizzazione di una memoria a byte o a word.

In esso sono rappresentati due banchi di memoria, di 32 Kbyte, per cui sono necessari 15 bit di indirizzo: allo scopo si utilizzano quelli più significativi $A_1 \div A_{15}$.
Il processore deve fornire un ulteriore segnale di controllo per indicare che si tratta di una operazione riguardante un byte oppure una parola.
In fig. 2.36 i due banchi sono abilitati da una combinazione di tale segnale con il bit Ao; se si vuole accedere ad una parola il segnale BYTE/WORD è a livello logico basso per cui entrambi i banchi di memoria ricevono il segnale di selezione CS/ ed inviano quindi complessivamente nel bus dei dati i 16 bit richiesti.

Se invece tale segnale è alto significa che il microprocessore vuole accedere ad un byte e il banco di memoria interessato sarà individuato dal valore del bit A_0: se esso vale 0 significa banco pari, altrimenti dispari. Si noti che per una organizzazione di memoria a parole si utilizzano le normali memorie a byte finora viste.

2.10. Ampliamento della memoria indirizzabile da un microprocessore

In un processore con 16 bit di indirizzo è possibile accedere al massimo a 65536 posizioni di memoria (64 K).
In qualche applicazione potrebbe rendersi necessario disporre di una maggior quantità di posizioni: ciò è possibile se si utilizza una particolare organizzazione hardware e se inoltre si adottano delle opportune avvertenze nel software. In fig. 2.37 è riportato un possibile schema che permette di espandere il numero di locazioni di memoria sino a 8 Mbyte.
L'area di memoria è suddivisa in pagine da 32 Kbyte ciascuna, per cui sono necessari solo 15 bit per individuarne una loro qualsiasi locazione: allo scopo si utilizzano i bit A_0-A_{14} del bus degli indirizzi come é indicato in fig. 2.37.

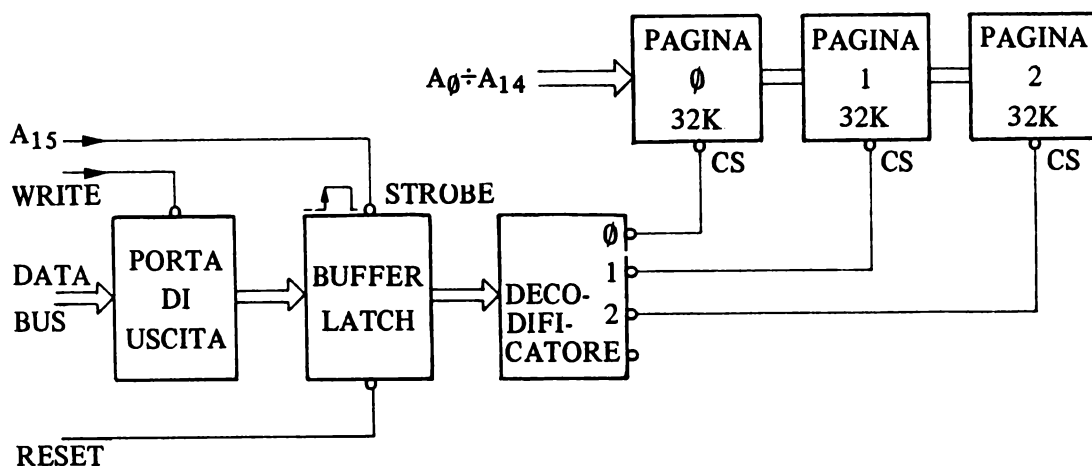



Fig. 2.37
Espansione della memoria in un microprocessore con 16 bit di indirizzo.

Il bus dei dati è collegato all'ingresso di un registro ad 8 bit che costituisce una porta di uscita del sistema a microprocessore. Questo registro può es-

sere indirizzato tramite una istruzione di uscita e quindi può memorizzare l'informazione inviatagli dal microprocessore non appena questi gli attiva l'ingresso WRITE.
Le uscite del primo registro sono collegate agli ingressi di un secondo registro, il quale acquisisce e memorizza i dati presenti al suo ingresso su comando tramite un segnale di STROBE che deriva dalla linea A_{15} del bus degli indirizzi.
All'accensione del sistema deve essere fornito a tale registro un segnale di RESET con il quale tutte le sue uscite si portano al valore logico basso: allora il decodificatore ad esso collegato attiva la sua uscita 0, la quale fa sì che l'unico banco di 32 Kbyte di memoria che può essere selezionato è quello di pagina 0.
Per passare a lavorare su un'altra pagina si deve agire a livello software: occorre innanzi tutto inviare sulla porta di uscita il numero della nuova pagina. Affinchè la successiva istruzione faccia riferimento a tale pagina è necessario che l'indirizzo della locazione a cui si vuole accedere abbia $A_{15}=1$ per cui l'indirizzo di tale istruzione è l'indirizzo effettivo, entro i 32 K, aumentato di 8000H. In questo modo il bit A_{15} vale 1 e rende disponibile al decodificatore quanto è presente sul primo registro e quindi si ha la selezione del banco desiderato. Si noti che l'operazione di aggiungere 8000H all'indirizzo è necessaria solamente per il passaggio da una pagina ad un'altra e non mentre si è in una stessa pagina nella quale si lavora normalmente utilizzando solamente i primi 15 bit dell'indirizzo. Si può ad esempio organizzare il programma per il cambio pagina nel seguente modo

```
(pagina vecchia) MVI   A, n. pagina nuova
                 OUT   Porta
                 JMP   (ADDRESS + 8000 H)

(si è nella nuova pagina)
```

2.11. Specifiche dei dispositivi di memoria

Per quanto riguarda le caratteristiche di velocità di risposta che deve presentare un dispositivo di memoria utilizzabile in un microelaboratore, esse possono essere desunte dalle caratteristiche dinamiche del microprocessore stesso.
Questi accede alla memoria per compiere tre operazioni distinte: quelle di lettura e di scrittura dei dati, e quella di fetch, cioè di lettura di una istruzione. Anche se il fetch è essenzialmente una operazione di lettura, essa avviene generalmente con modalità diverse rispetto a quest'ultima per cui conviene considerarlo distinto. Ad esempio nel microprocessore Z80 mentre il ciclo di lettura di una dato in memoria impone che quest'ultima fornisca i dati entro due periodi e mezzo del segnale di clock del micropro-

cessore, in un ciclo di fetch i dati debbono essere forniti entro due soli periodi di clock. E' evidente che le caratteristiche necessarie alla memoria nei due casi sono abbastanza diverse e che una memoria che contiene codici operativi deve possedere una velocità di risposta maggiore rispetto ad una che contenga solo dati.
Per quanto riguarda una memoria di tipo ROM, occorrerà analizzare in dettaglio i tempi necessari perché essa esegua l'operazione di lettura, mentre per una memoria RAM occorrerà determinare anche quelli relativi ad operazioni di scrittura.

2.11.1. *Memoria ROM*

Si vogliono ora determinare le caratteristiche dinamiche necessarie ad una memoria ROM interessata a cicli di fetch da parte di un microprocessore Z80.
Per esso il diagramma temporale relativo ad un ciclo di fetch è indicato in fig. 2.38, il segnale di riferimento dei tempi è il clock ϕ e sul fronte di salita del primo periodo, T_1, inizia il ciclo in oggetto.

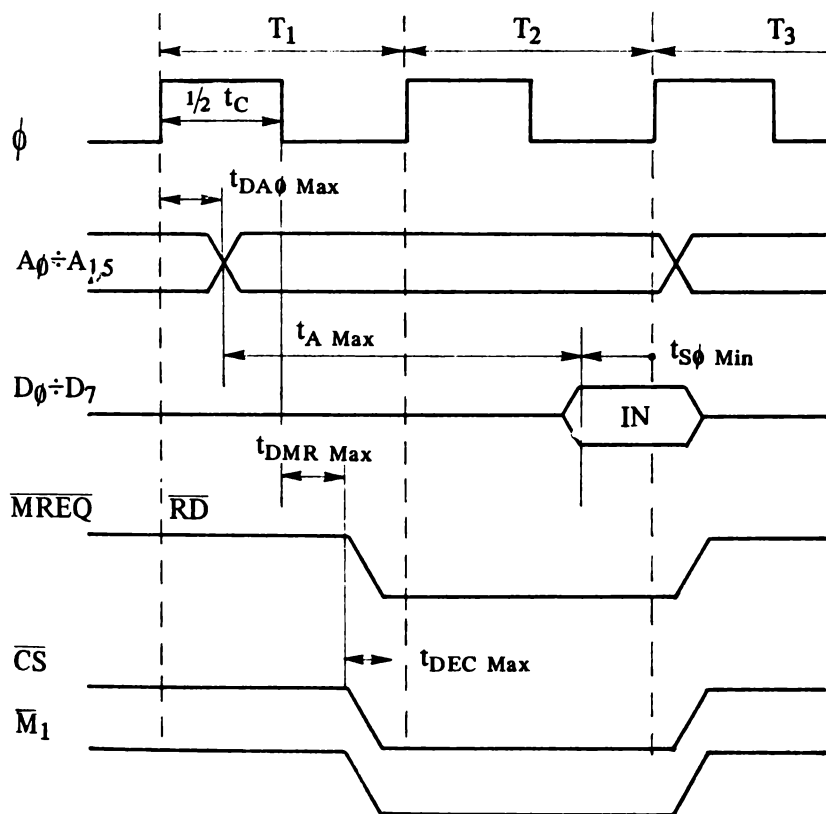



Fig. 2.38
Ciclo di fetch del microprocessore Z80: tempi caratteristici.

Dopo un certo tempo, $t_{DA\phi}$, dall'istante di riferimento, il microprocessore pone sul bus degli indirizzi quello che individua la cella di memoria che deve essere letta e, successivamente, dopo un tempo t_{DMR} dal fronte di discesa del segnale di clock in T_1, sono resi attivi i segnali di controllo MREQ/ e RD/ mediante i quali il microprocessore specifica che l'operazione che si appresta ad eseguire è di lettura di memoria. Il segnale M_1/ indica inoltre che si tratta di una operazione di fetch.
In condizioni normali il microprocessore procede alla lettura del dato sul fronte di salita del terzo periodo di clock T_3: entro tale istante la memoria deve avere posto sul bus dei dati l'informazione richiesta.
La memoria sarà connessa al bus degli indirizzi ed a quello di controllo per il tramite di un circuito di decodifica degli indirizzi che le fornirà il segnale di selezione CS/; si supponga per ora che tale circuito non introduca dei ritardi: si vuole determinare quale è il valore massimo del tempo di accesso, t_{AMAX}, che può presentare la memoria.
La parte del diagramma temporale che qui interessa è riportato in fig. 2.39: in essa sono indicati il ritardo $t_{DA\phi}$ dopo il quale gli indirizzi sono validi e quello $t_{S\phi}$, di set up, necessario affinché i dati presenti sul bus siano acquisiti correttamente dal microprocessore.

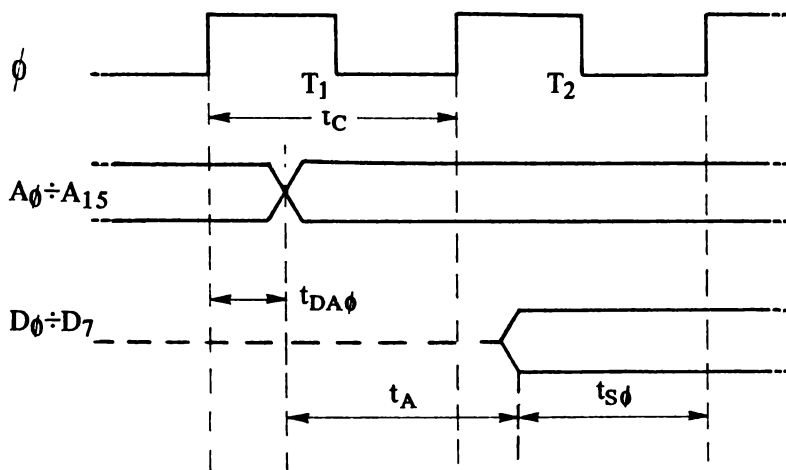



Fig. 2.39
I tempi caratteristici nel ciclo di lettura del microprocessore Z80.

Le caratteristiche fornite per il microprocessore indicano qual'è il ritardo massimo $t_{DA\phi MAX}$ e quale l'intervallo di tempo minimo $t_{S\phi_{min}}$ ammesso.

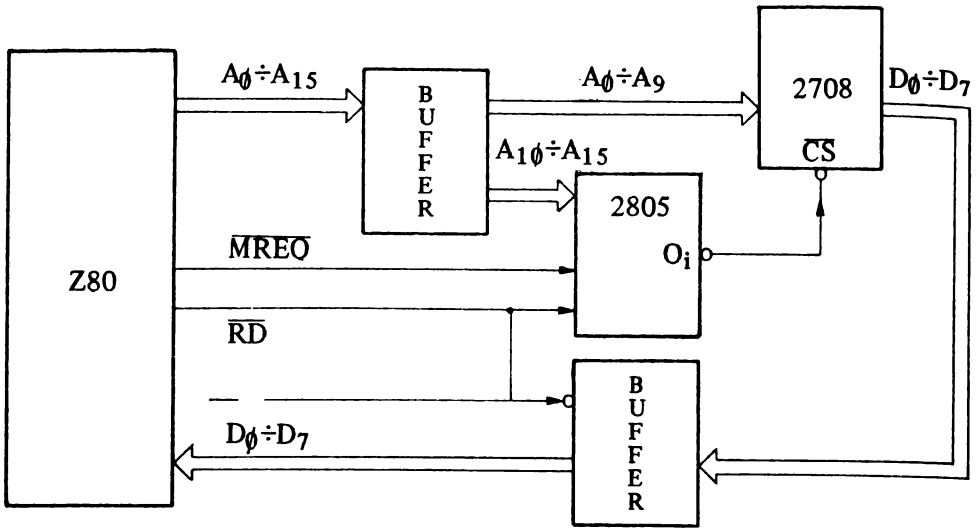



Fig. 2.40 a
Schema per la verifica della compatibilità della memoria EPROM 2708 con il microprocessore Z80.

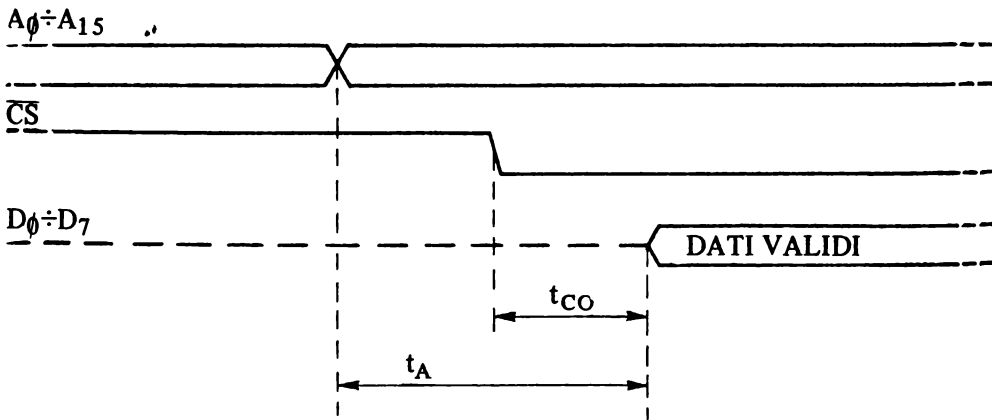



Fig. 2.40 b
Particolare delle caratteristiche dinamiche della EPROM 2708.

Evidentemente il tempo minimo di accesso disponibile per la memoria è dato da:

$$t_{A_{min}} = 2 \cdot t_c - t_{DA\phi_{MAX}} - t_{S\phi_{min}} \qquad (2\text{-}1)$$

ove t_c è il periodo del clock ϕ.
Finora non si è tenuto conto del fatto che la memoria abbisogna di un certo tempo, t_{co}, dall'istante di attivazione del suo ingresso di controllo, CS/, per presentare i dati stabiliti in uscita.
Se tale CS/ è fornito, come si vede in fig. 2.40a da un decodificatore (del tipo 8205) ai cui ingressi sono inviate parte delle linee del bus degli indirizzi e le due linee di controllo MREQ/ e RD/. il diagramma temporale che interessa in questo caso è riportato in fig. 2.40b. Il segnale CS/ sarà attivo dopo un tempo, $t_{DEC_{MAX}}$, dall'istante in cui sono attive le linee MREQ/ e RD/, a causa della velocità di risposta finita del decodificatore.
Rispetto al fronte di salita del segnale di clock in T_1 i segnali di controllo MREQ/ e RD/ sono attivati con un ritardo massimo dato da:

$$t_{DMR\phi_{MAX}} = 1/2\ t_c + t_{DMR_{MAX}}$$

per cui il tempo minimo di abilitazione, t_{co}, a disposizione per la memoria risulta pari a:

$$t_{co_{min}} = 2t_c - t_{DMR\phi_{MAX}} - t_{DEC_{MAX}} - t_{S\phi_{min}} \qquad (2\text{-}2)$$

I valori forniti dalla (2-1) e dalla (2-2) permettono di specificare le caratteristiche dinamiche che deve avere una memoria soggetta a soli cicli di fetch, che, come si è già fatto notare, sono quelli che impongono vincoli temporali più stretti se deve essere impiegata in un microelaboratore che come unità centrale usa lo Z80.
Se si volesse, ad esempio, utilizzare una ROM del tipo 2708, della quale le caratteristiche che qui interessano sono indicate in fig. 2.41, occorre calcolare i valori forniti dalle (2-1) e (2-2) e verificare la compatibilità con quelli delle 2708.
Per il microprocessore Z80 funzionante con t_c = 400 ns risultano, con $t_{DEC_{MAX}}$ = 20 ns,

$$t_A = 800 - 160 - 50 = 590 \text{ ns}$$

$$t_{co} = 800 - 200 - 100 - 50 - 20 = 430 \text{ ns}$$

che sono molto maggiori rispettivamente a $t_{A_{MAX}}$ = 350 e $t_{co_{MAX}}$ = 120 ns necessari alla ROM.

Symbol	Parameter	2708-1 Limits Min.	Typ.	Max.	2708, 2708L, Limits Min.	Typ.	Max.	Units
t_{ACC}	Address to Output Delay		280	350		280	450	ns
t_{CO}	Chip Select to Output Delay		60	120		60	120	ns
t_{DF}	Chip Deselect to Output Float	0		120	0		120	ns
t_{OH}	Address to Output Hold	0			0			ns

CAPACITANCE[1] $T_A = 25°C$, $f = 1$ MHz

Symbol	Parameter	Typ.	Max.	Unit.	Conditions
C_{IN}	Input Capacitance	4	6	pF	$V_{IN} = 0V$
C_{OUT}	Output Capacitance	8	12	pF	$V_{OUT} = 0V$

Note: 1. This parameter is periodically sampled and is not 100% tested.

A.C. TEST CONDITIONS:

Output Load: 1 TTL gate and $C_L = 100$ pF
Input Rise and Fall Times: ≤20 ns
Timing Measurement Reference Levels: 0.8V and 2.8V for inputs; 0.8V and 2.4V for outputs.
Input Pulse Levels: 0.65V to 3.0V

Waveforms

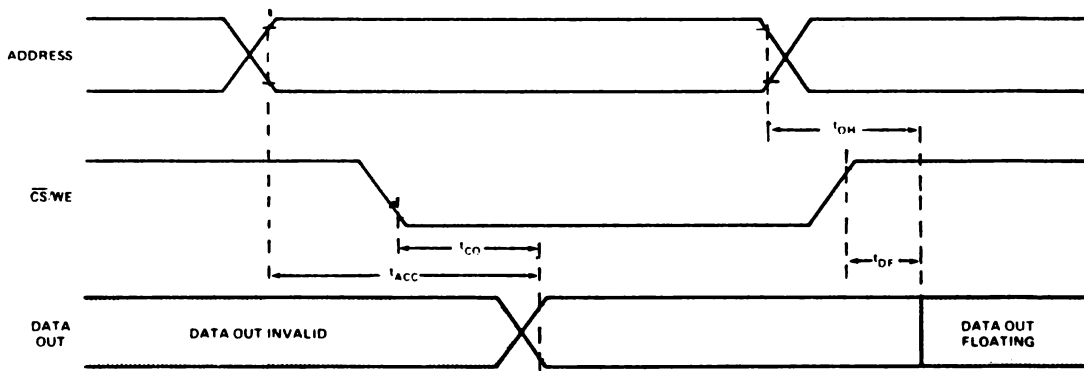



Fig. 2.41
Caratteristiche dinamiche della memoria EPROM 2708 (Intel Component Data Catalogo 1979)

Nei tempi di (590 – 350) = 240 ns e (430 – 120) = 310 ns dovranno essere compresi tutti i ritardi di propagazione attraverso gli eventuali buffer presenti nel sistema e attraverso le stesse linee dei bus, ritardi che finora non si sono presi in considerazione.

2.11.2. *Inserimento di cicli di attesa*

Si può anche verificare il caso che i tempi di funzionamento di una certa memoria, scelta magari per il minor costo rispetto ad altre o anche solo per questioni di pronta disponibilità, non soddisfino le relazioni prima ricavate. Per ovviare a tale inconveniente si possono seguire due metodi: nel primo si diminuisce la frequenza di funzionamento del microprocessore; nel secondo si fa in modo che il microprocessore conceda più tempo alla memoria per fornire i dati richiesti. La prima soluzione, estremamente semplice in quanto si tratta solamente di variare alcuni componenti nel circuito che genera il segnale di clock, presenta tuttavia il notevole inconveniente di diminuire la velocità di tutte le attività del microprocessore e quindi risultano peggiorate le prestazioni dell'intero sistema. L'altra soluzione, che consiste nell'inserire degli stati di attesa nel funzionamento dell'unità centrale quando ciò si renda necessario, peggiora solo di poco le caratteristiche di velocità dell'elaboratore; si noti che quasi tutti i microprocessori hanno la possibilità, tramite un segnale di ingresso denominato a seconda dei casi WAIT oppure READY, di aumentare la durata di un ciclo di lettura, o di scrittura, di un numero intero di periodi di clock.
In genere è sufficiente aggiungere qualche periodo di clock per adeguare la velocità di funzionamento dell'unità centrale con quella della memoria, oppure di altri componenti più lenti.
Si può allora utilizzare un circuito che renda attivo il segnale di WAIT, nel caso si utilizzi il microprocessore Z80, ogni qualvolta esso deve accedere alla memoria per una lettura. Lo schema di tale circuito potrebbe, ad esempio, essere quello riportato nella fig. 2.42.

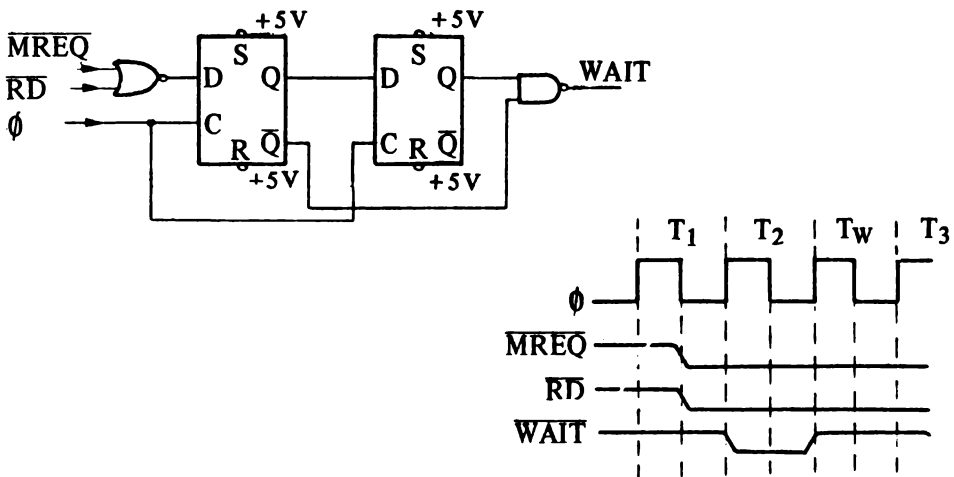



Fig. 2.42
Schema per l'inserzione di uno stato di wait per ogni operazione di lettura da memoria

Sempre nel caso nell'impiego dello Z80 si ricordi che l'accesso alla memoria avviene con modalità distinte a seconda che si tratta di una lettura di un codice operativo, oppure di un dato. Nel primo caso infatti la lettura avviene sul fronte di discesa sempre dello stesso periodo, per cui si ha a disposizione in più un tempo pari ad un semiperiodo. Per questo motivo può essere conveniente imporre l'intervallo di Wait solamente durante la lettura del codice operativo, migliorando in tal modo le prestazioni dell'intero sistema. Per far ciò si può, ad esempio, adottare il circuito di fig. 2.43.

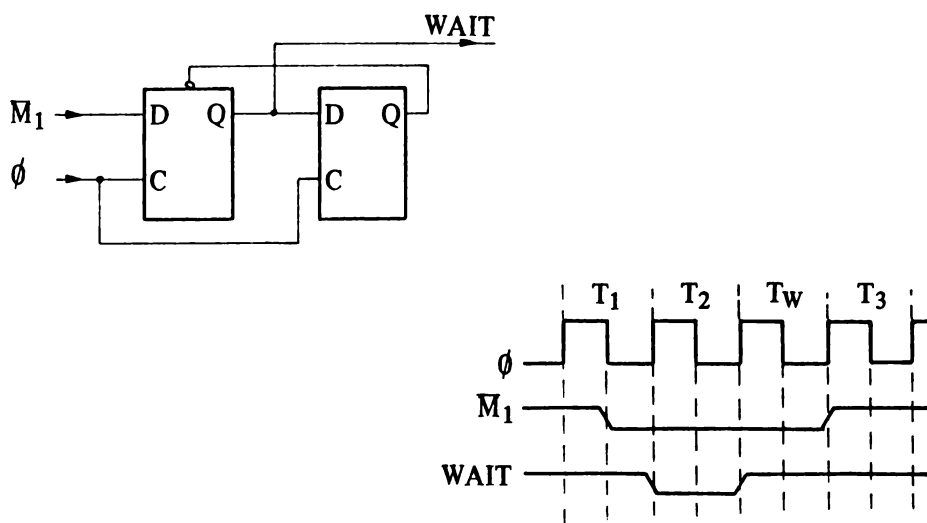



Fig. 2.43
Schema per l'inserzione di uno stato di Wait nella lettura di codice operativo

2.11.3. *Memorie RAM*

Per quanto riguarda l'utilizzazione di una memoria RAM in un microelaboratore, la verifica dei tempi di risposta della stessa ai comandi del microprocessore deve essere effettuata sia per il ciclo di lettura che per quello di scrittura.

La determinazione del tempo di accesso, t_A, e del tempo di abilitazione, t_{co}, nel caso di un ciclo di lettura utilizza relazioni analoghe a quelle che si sono trovate nel caso del ciclo di fetch, con l'avvertenza che ora il tempo messo a disposizione dal microprocessore può essere diverso. Ad esempio, per il microprocessore Z80, la memoria deve fornire i dati entro il fronte di discesa dell'impulso di clock del periodo T_3. Le relazioni (2-1) e (2-2) diventano allora:

$$t_{A_{min}} = 2,5\ t_c - t_{DA_{MAX}} - t_{S\phi_{min}} \tag{2-3}$$

$$t_{co_{min}} = 2,5\ t_c - t_{DMR\phi_{MAX}} - t_{s\phi_{min}} - t_{DEC_{MAX}} \tag{2-4}$$

Se si volesse utilizzare una memoria RAM del tipo 2102A-4 per la quale, in un ciclo di lettura, si ha $t_{A_{MAX}} = 450$ ns; $t_{co_{MAX}} = 230$ ns, in uno schema analogo a quello indicato in fig. 2.40, dato che ora le (2-3) e (2-4) forniscono i valori rispettivamente di 790 e 630 ns, restano a disposizione per i ritardi di propagazione attraverso i vari buffer e le linee dei bus del sistema, dei tempi di (790 − 450) = 340 ns per il tempo di accesso e di (630 − 230) = 400 ns per quello di abitazione.
Per quanto riguarda invece un ciclo di scrittura di una memoria, i tempi che lo caratterizzano sono indicati in fig. 2.44; si deve notare che l'impulso di segnalazione di scrittura non può avere una durata inferiore ad un certo valore $t_{W_{min}}$; inoltre gli indirizzi ed i dati devono essere forniti con un certo anticipo e cioè con dei tempi di set-up, t_{sa} e t_{sd} rispettivamente, non inferiori a prefissati valori minimi. In genere è richiesto che i dati siano presenti anche dopo che il segnale WR/ è disattivato (tempo di hold t_{hd}). Dalle caratteristiche dinamiche fornite per un microprocessore sono sempre ricavabili gli intervalli di tempo che ora interessano; essi sono indicati in fig. 2.45, ove i simboli impiegati sono gli stessi usati per la memoria con l'aggiunta di un pedice p per indicare che si riferiscono alle caratteristiche del processore.

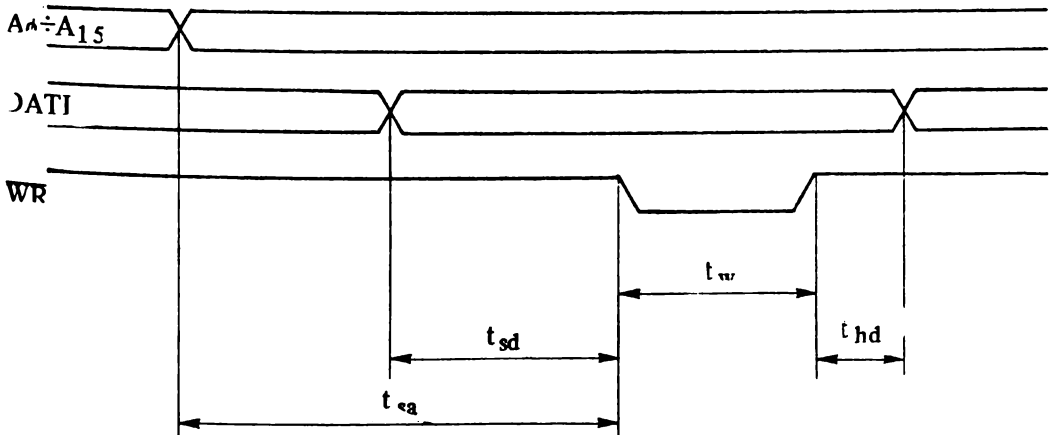



Fig. 2.44
Ciclo di scrittura di una memoria: tempi caratteristici.

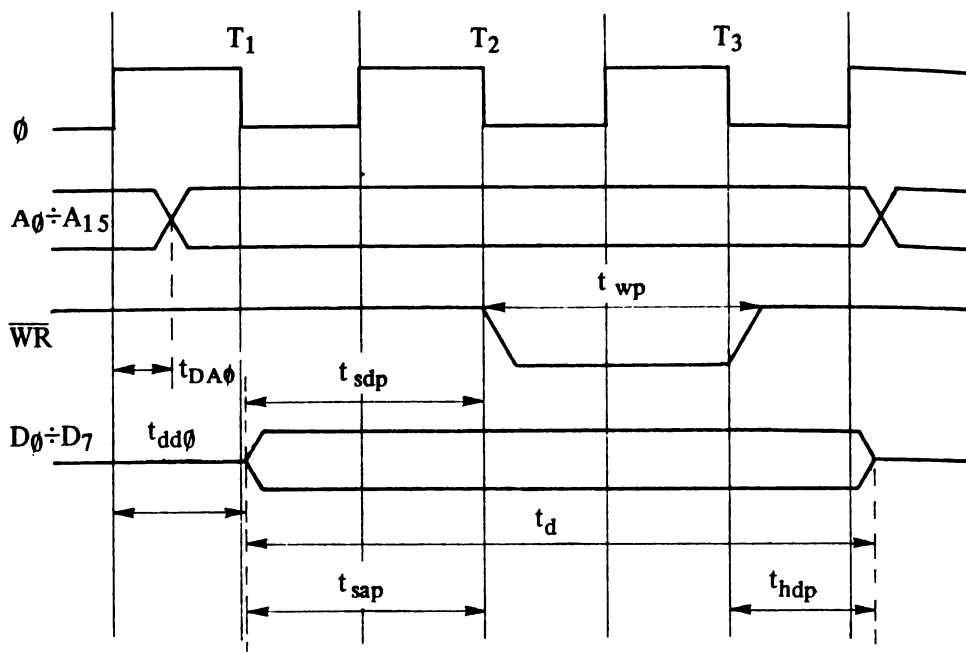



Fig. 2.45
Ciclo di scrittura in memoria nel microprocessore Z80.

Il microprocessore deve presentare delle caratteristiche tali che sicuramente siano soddisfatte le specifiche temporali della memoria e cioè:

$$t_{sap_{min}} \geqslant t_{sa_{min}} \qquad a)$$

$$t_{sdp_{min}} \geqslant t_{sd_{min}} \qquad b)$$

$$t_{wp_{min}} \geqslant t_{w_{min}} \qquad c) \qquad (2\text{-}5)$$

$$t_{hd_{min}} \geqslant t_{hd_{min}} \qquad d)$$

$$t_{dp_{min}} \geqslant t_{sd_{min}} + t_{w_{min}} + t_{hd_{min}} \qquad e)$$

Se si fa l'ipotesi di utilizzare una memoria 8102-A4 per essa risulta

$t_{sa_{min}} = 20$ ns

$t_{sd_{min}} = 0$ ns ($t_w - t_d = 300 - 300$)

$t_{w_{min}} = 300$ ns

$t_{hd_{min}} = 0$ ns

$t_{d_{min}} = 300$ ns

Mentre per il microprocessore Z80 risulta, a 400 ns di periodo di clock,

$t_{sap_{min}} = 1,5\ t_c + t_{WR\phi_{min}} - t_{ap_{MAX}} = 600 + 90 - 160 = 530$ ns

$t_{sd_{min}} = 400 - 180 = 220$ ns

$t_{wp_{min}} = 400 - 40 = 360$ ns

$t_{hdp_{min}} = 180$ ns

$t_{dp_{min}} > 800$ ns

e quindi le (2-5) sono ampiamente soddisfatte.
Finora la verifica della compatibilità di una memoria RAM con lo Z80 è stata verificata senza tener conto del fatto che essa deve essere abilitata: la 8102-A4 ha bisogno di un impulso di CS/ che, con riferimento all'istante di attivazione di WR, deve essere presente almeno per un tempo, t_{cw}, non inferiore a 300 ns.
Poichè per il microprocessore Z80 i segnali di MREQ/ e di RD/, e WR/, utilizzati per effettuare la scrittura, sono attivati almeno 1,5 periodi di clock prima che esso disattivi il segnale WR/, allora anche questa condizione è soddisfatta ed addirittura si hanno a disposizione 1,5 x 400 – 300 = = 300 ns almeno per gli eventuali tempi di ritardo di propagazione attraverso i soliti buffer e le linee dei bus.

2.12. Gli ingressi di controllo in una memoria

Gli ingressi di controllo presenti in un circuito integrato di memoria possono variare sia come tipo che come numero a seconda e della memoria e del costruttore della stessa.

Sono sempre presenti uno o più ingressi di abilitazione; nomi tipici per essi sono: CE (chip enable) oppure CS (chip select) e la loro funzione è quella di dare inizio alle attività del dispositivo, anche se con modalità che possono essere leggermente diverse. Esistono delle memorie che presentano entrambi gli ingressi CE e CS. In tal caso in genere il segnale di CE fa passare la memoria dallo stato di attività a quella di standby e viceversa, se tale modo di funzionamento è previsto, mentre l'ingresso CS permette di attivare le varie operazioni sulla cella di memoria interessata. In molti casi il significato dei due ingressi è equivalente.
In una memoria possono essere contemporaneamente presenti più ingressi di CS e questi possono essere attivi o a livello logico alto o a quello basso; con l'aumentare del loro numero in genere si possono semplificare i circuiti esterni di decodifica degli indirizzi, come si è già visto nell'esempio di fig. 2.16.
Le uscite dei dati delle memorie presentano la caratteristica di essere in grado di assumere uno stato di alta impedenza, e solo raramente sono del tipo open collector. Il passaggio nello stato di alta impedenza può essere automatico a seconda della attività in quel momento esplicata dalla memoria stessa; ad esempio, in fase di lettura, una volta selezionata, con l'attivazione dell'ingresso CS, la memoria automaticamente si porta nello stato di uscita a bassa impedenza per mettere a disposizione i dati richiesti, e ritorna in quello di alta impedenza una volta cessata l'attivazione di cui sopra. Per alcuni tipi di memoria è possibile imporre con un comando che il buffer di uscita si ponga nello stato normale o in quello ad alta impedenza; tale ingresso di abilitazione delle uscite prende di solito il nome di OE (output disable). Nelle memorie in cui è presente questo ingresso di controllo, una volta richiesto un dato (tramite gli ingressi di indirizzo, i segnali di controllo CS e di attivazione dell'operazione di lettura) esse procedono alla selezione della cella richiesta e ne inviano il contenuto al buffer di uscita, il quale però non lo rende disponibile all'esterno se non viene attivato il segnale OE. L'utilità di tale ingresso di controllo sarà illustrata fra breve. Anche gli ingressi che impongono alla memoria l'operazione di lettura o di scrittura che si intende compiere, possono essere presenti in numero diverso: ad esempio in una memoria ROM, essendo unica l'operazione su essa eseguibile, molto spesso si trova solo l'ingresso di CS. In una memoria RAM invece è ovviamente indispensabile specificare il tipo di operazione da eseguire: in qualche caso è presente l'ingresso R/W con il quale, a seconda del valore impostogli si dà tale indicazione; in altri è presente un ingresso detto WE (Write Enable) con significato analogo.
Nelle memorie RAM dinamiche sono presenti anche altri ingressi di controllo la cui funzione e significato verranno illustrati nel capitolo ad esse dedicato.
Si descrivono ora alcuni metodi con cui si possono generare i segnali da

applicare agli ingressi di controllo dei vari tipi di memoria, utilizzando le informazioni presenti nei bus di un microelaboratore.
I casi tipici descritti riguardano un dispositivo di memoria ROM che abbia disponibile solo l'ingresso di selezione, CS/, uno di RAM con in più l'ingresso di R/W ed infine una RAM che presenti anche l'ingresso di controllo di abilitazione delle uscite OE.

2.12.1. *Memorie ROM con il solo ingresso CS*

Si voglia ad esempio realizzare un banco di memoria ROM utilizzando quattro circuiti integrati, ognuno contenente 256 locazioni, di 8 bit, nei quali è presente il solo ingresso di controllo CS/.
Un possibile schema per la selezione potrebbe essere quello riportato nella fig. 2.46.

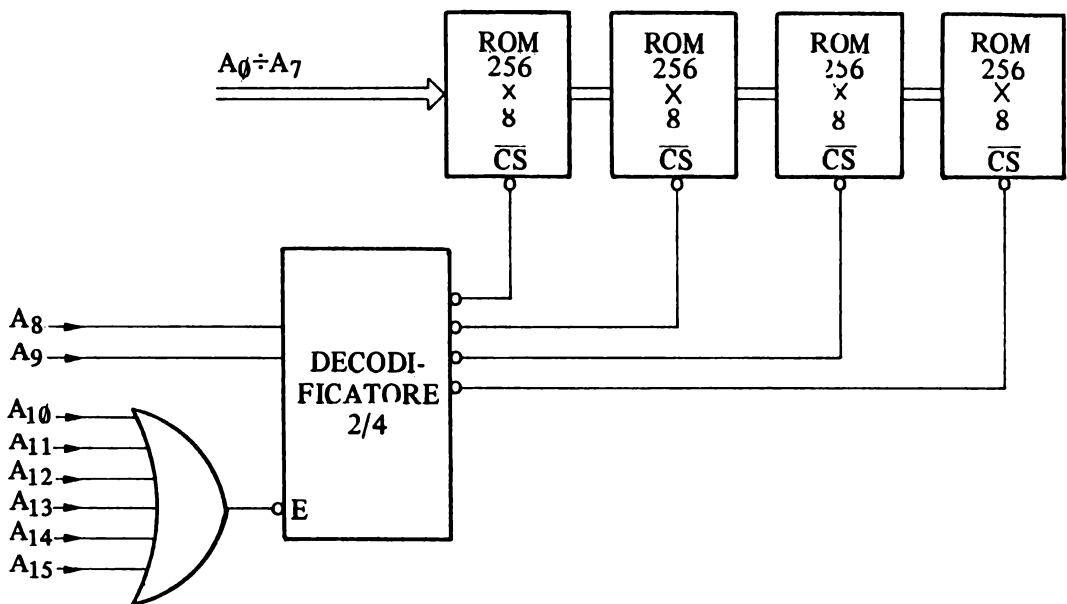



Fig. 2.46
Selezione di memorie ROM dotate del solo comando CS/ (possibilità di bus contention).

Se le memorie utilizzate non presentano particolari caratteristiche dinamiche tale schema può dare luogo a degli inconvenienti.

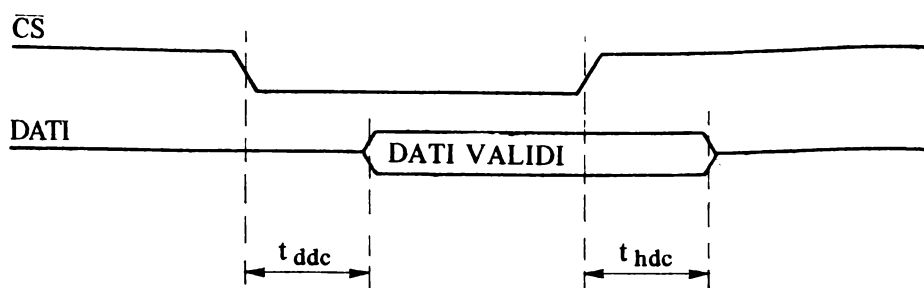



Fig. 2.47
Caratteristiche dinamiche del ciclo di lettura di una memoria ROM.

Nella fig. 2.47 è indicato il diagramma temporale relativo a questo tipo di memorie, dove sono messi in evidenza i due intervalli di tempo che in questo caso interessano:

- t_{ddc} = tempo di ritardo, che intercorre dall'istante di attivazione dell'ingresso CS/ a quello in cui i dati sono disponibili in uscita;
- t_{hdc} = tempo in cui permangono i dati in uscita una volta disattivato l'ingresso CS/.

All'infuori dell'intervallo di tempo in cui i dati sono disponibili, si deve intendere che l'uscita si trova nello stato di alta impedenza.
Se il comando di CS/, relativo ad una memoria, è attivato contemporaneamente alla disattivazione di un altro, si possono verificare due situazioni diverse a seconda dei valori di t_{ddc} e di t_{hdc}.
Nel primo dei due casi, riportati in fig. 2.48, ove $t_{ddc_B} > t_{hdc_A}$ in ogni istante è una sola memoria che controlla il bus dei dati; nel secondo invece, ove $t_{ddc_B} < t_{hdc_A}$, il bus dei dati è pilotato contemporaneamente dalla memoria A e da quella B.
Nell'intervallo tratteggiato in fig. 2.48, di durata $(t_{hdc_A} - t_{ddc_B})$, è allora indeterminato il contenuto del bus dei dati: si dice che si è verificato un conflitto sul bus (bus contention). Se qualche dispositivo preleva i dati in quell'intervallo di tempo ovviamente esso acquisisce una informazione errata.
In tali situazioni si può provocare un danneggiamento dei componenti collegati al bus; a volte può solo verificarsi la generazione di disturbi, dovuti alle elevate correnti in gioco, i quali però determinano un funzionamento

anomalo dell'intero microelaboratore con notevoli difficoltà per indivuarne la causa.

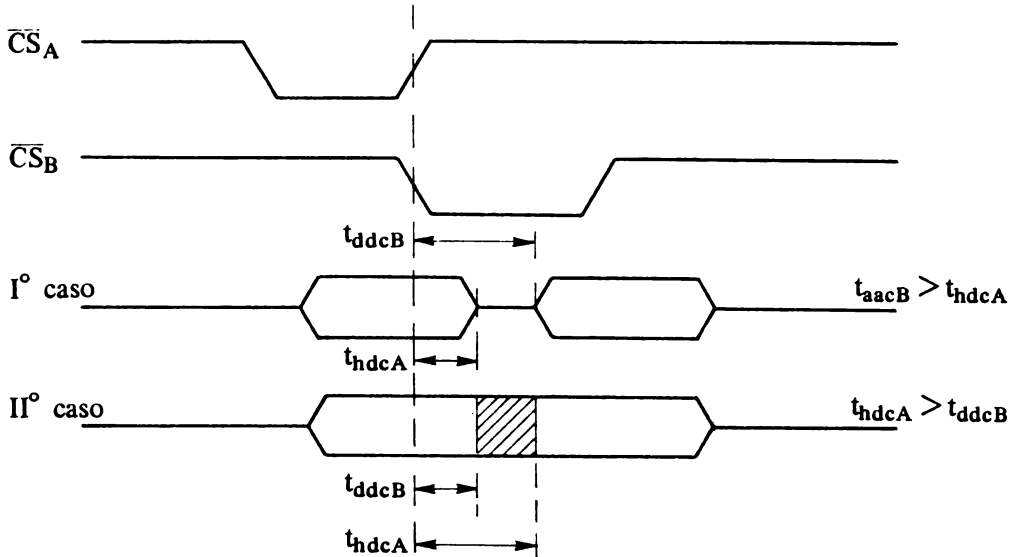



Fig. 2.48
Possibilità di bus contention.

E' evidente allora che, come in fig. 2.46, se si utilizzano per la selezione di una memoria solamente i segnali del bus degli indirizzi e si hanno ad esempio più cicli macchina successivi che eseguono l'operazione di lettura si può avere un problema di bus contention. Per evitare tale inconveniente si debbono utilizzare in modo opportuno altri segnali di controllo forniti dal microprocessore.
Le soluzioni che si possono adottare dipendono e dal tipo di memoria e dal microprocessore utilizzato: supponendo di impiegare lo Z80 si può utilizzare il circuito di fig. 2.49.
Il decodificatore procede alla decodifica solamente se è attivo il suo ingresso di selezione EN/, il quale si troverà in tale condizione solo quando sono contemporaneamente attivi i due segnali MREQ/ e RD/. Da fig. 2.50 si può notare che il segnale EN/ sarà attivato e disattivato sul fronte di discesa dei periodi di clock T_1 e T_3 rispettivamente; in questo modo nel caso di due cicli macchina di lettura consecutivi, si ha la disattivazione di tutte le memorie per almeno un periodo di clock.

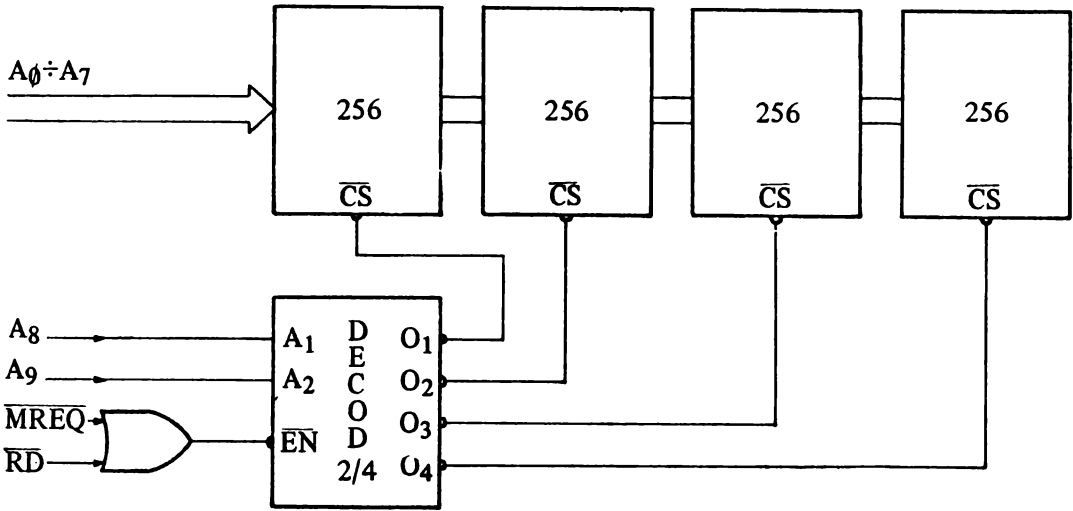



Fig. 2.49
Selezione della memoria con eliminazione del bus contention.

MEMORY READ OR WRITE CYCLES

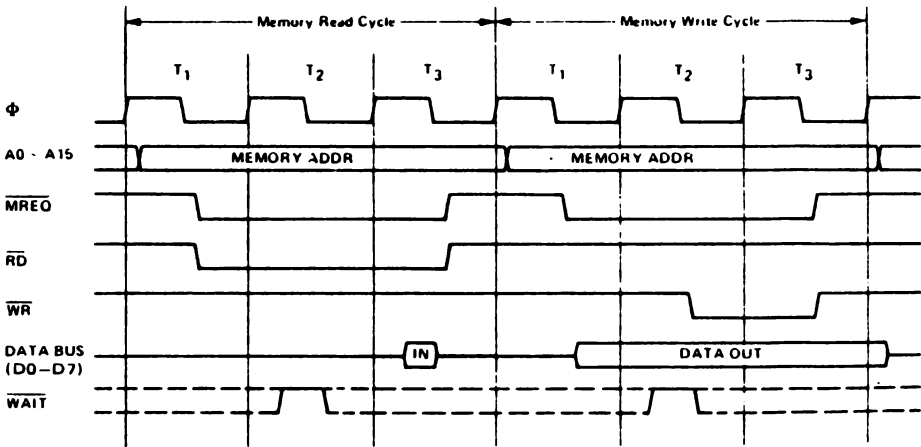



Fig. 2.50
Ciclo di lettura e scrittura in memoria nel microprocessore Z80 (Mostek Z80 Microcomputer Data Book 1981)

In realtà bisognerebbe procedere ad un calcolo più preciso in quanto non sono stati tenuti presenti i vari ritardi introdotti dal decodificatore, dalla porta OR, dalle linee del bus e dai buffer eventualmente presenti; tuttavia questi ritardi complessivamente saranno certamente minori del periodo di clock, t_C, del microprocessore.
Un'altra soluzione al problema del bus contention è quella di utilizzare delle memorie che presentano anche l'ingresso di controllo OE/. In questo caso il circuito che può essere utilizzato è quello riportato nella fig. 2.51 dove l'uscita è tolta dallo stato di alta impedenza solamente quando sono contemporaneamente attivi i due segnali MREQ/ e RD/.

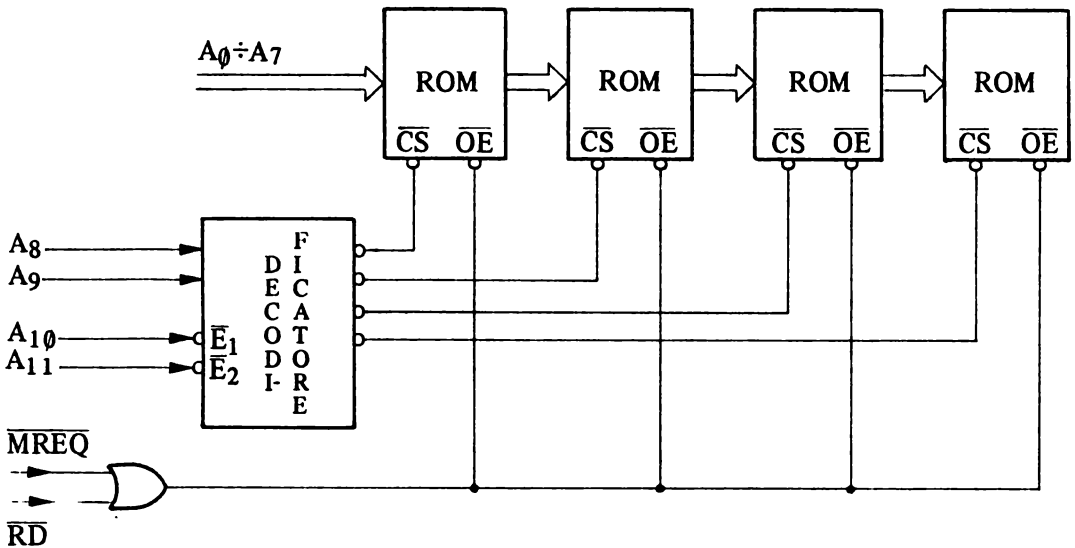
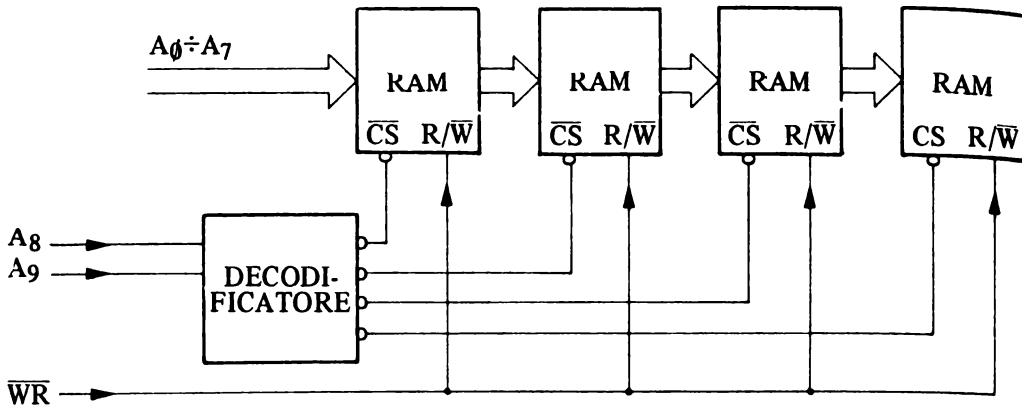



Fig. 2.51
Utilizzazione del comando di abilitazione delle uscite.

2.12.2. *Memorie RAM con ingressi CS e R/W.*

Si prenda ora in esame l'utilizzazione di una memoria RAM dotata dei soli ingressi di controllo CS/ e R/W e che abbia le linee dei dati bidirezionali.
Durante l'operazione di lettura si possono avere gli stessi problemi visti nel caso delle memorie ROM; nella scrittura se ne presentano altri che possono dar luogo al bus contention. Uno schema che viene spontaneo di adottare è indicato nella fig. 2.52 dove è anche riportata la tabella di verità relativa alla memoria.

$\overline{CS}$	R/$\overline{W}$	
0	0	Scrittura
0	1	Lettura
1	X	H.Z.

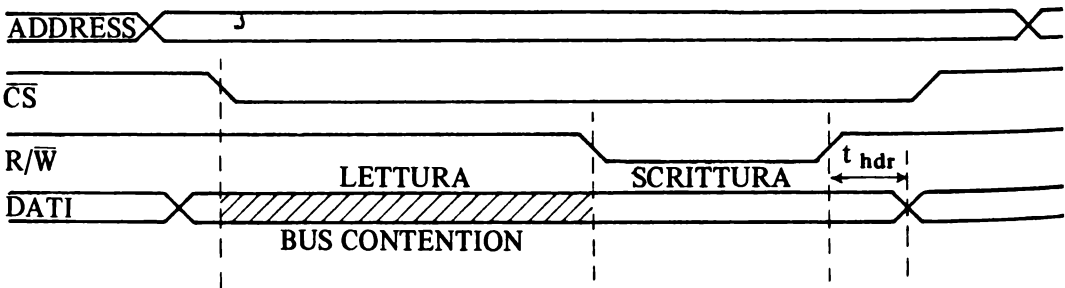



Fig. 2.52
Bus contention in RAM con un solo ingresso per il comando di lettura-scrittura.

Il circuito di fig. 2.52 presenta degli inconvenienti: infatti una volta stabilizzati gli indirizzi, dopo un certo ritardo, dipendente dal decodificatore, è attivato il segnale CS/. In questo istante la memoria si porta nelle condizioni di lettura, in quanto il segnale R/W è ancora a livello logico alto, e quindi essa pone sul bus dei dati la configurazione di bit contenuta nella

locazione individuata dall'indirizzo presente. Se contemporaneamente il microprocessore invia il dato da scrivere sulla memoria sono presenti due trasmettitori attivi per cui si verifica la situazione di bus contention. Tale situazione cessa solamente quando il segnale R/W si porta al livello basso per indicare l'operazione di scrittura.
Con l'uso di memorie RAM in un bus di dati bidirezionali si può verificare una situazione di bus contention anche nel caso che ad un ciclo di scrittura segua immediatamente un ciclo di lettura.
In fig. 2.53 è riportato il diagramma temporale relativo alle due situazioni.

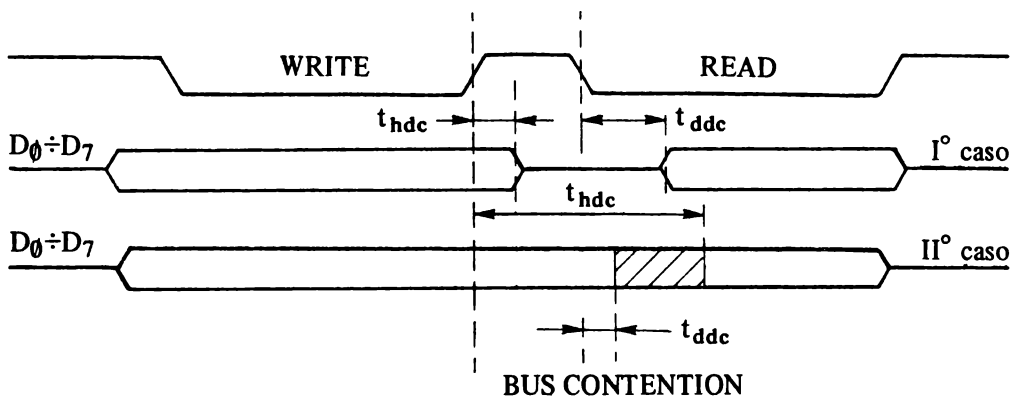



Fig. 2.53
Bus contention nel caso di un ciclo di lettura che segue quello di scrittura.

In una operazione di scrittura i dati devono essere mantenuti validi per un certo tempo dopo la disattivazione di CS/. Se immediatamente si fa seguire una operazione di lettura qualificata solamente da CS/, in quanto il segnale R/W viene lasciato al suo livello di riposo e cioè alto, si può verificare che una memoria particolarmente veloce sia in grado di inviare i propri dati sul bus quando in questo sono ancora presenti i dati della precedente fase di scrittura. Durante un certo intervallo di tempo si ha ancora la situazione di bus contention.
Questo può essere evitato se la memoria ha a disposizione anche l'ingresso OE/: in questo caso infatti, utilizzando un circuito come quello di fig. 2.51, certamente OE/ sarà attivato dal microprocessore un certo intervallo di tempo dopo che è stato portato a livello logico basso l'ingresso di CS/.

2.12.3. *Uso di memoria in sistemi a bus multiplexati*

Un'altra situazione in cui si può verificare il bus contention è quella che si

ha quando si utilizzano dei microprocessori con bus multiplexati.
Se, ad esempio, il bus dei dati è multiplexato con quello degli indirizzi, come accade nei microprocessori Z8000, 8085 ecc., in ogni ciclo di lettura da memoria il bus dei dati contiene, per un certo intervallo di tempo, informazioni relative all'indirizzo, fornite dal microprocessore; successivamente quest'ultimo ne rilascia il controllo per permettere alla memoria di fornire le informazioni richieste.
Se la memoria presenta una elevata velocità di risposta e se la sua attivazione è fornita solo attraverso l'ingresso di CS, può accadere che essa invii i dati richiesti quando sono presenti ancora sulle stesse linee gli indirizzi forniti dal microprocessore. Ciò è illustrato in fig. 2.54 ove per chiarezza sono indicati in modo distinto il contenuto dei bus dei dati indirizzo.

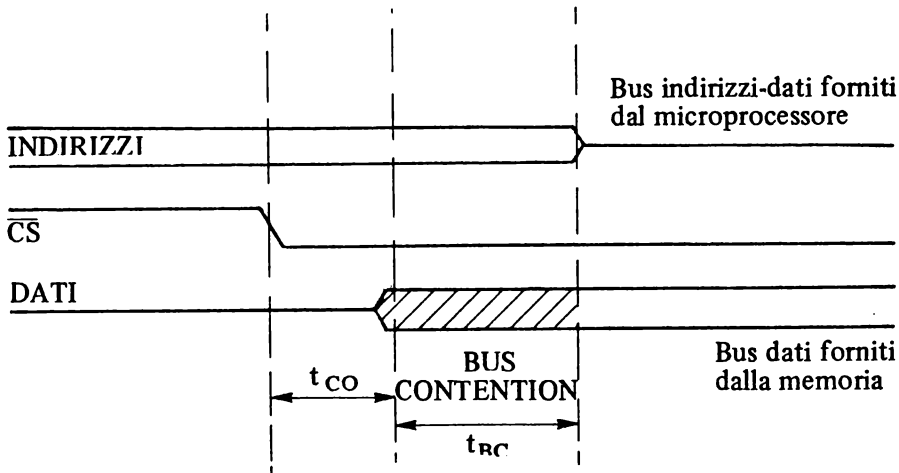



Fig. 2.54
Bus contention nel caso di bus dei dati e degli indirizzi multiplexati.

In essa si vede come esista un intervallo di tempo, t_{BC}, di bus contention, il quale risulta tanto più grande quanto minore è il tempo di risposta, t_{CO}, della memoria, all'attivazione dell'ingresso di abilitazione CS/.
Anche in questo caso si può ovviare all'inconveniente utilizzando una delle due tecniche precedentemente viste che consistono o nell'attivare l'unico ingresso di abilitazione della memoria solo quando è presente il segnale di lettura, RD/, fornito dal microprocessore, oppure nell'inviare tale segnale all'ingresso di abilitazione delle uscite, OE/, per le memorie che ne sono provviste. Le due soluzioni sono illustrate nelle fig. 2.55 a) e b) rispettivamente.

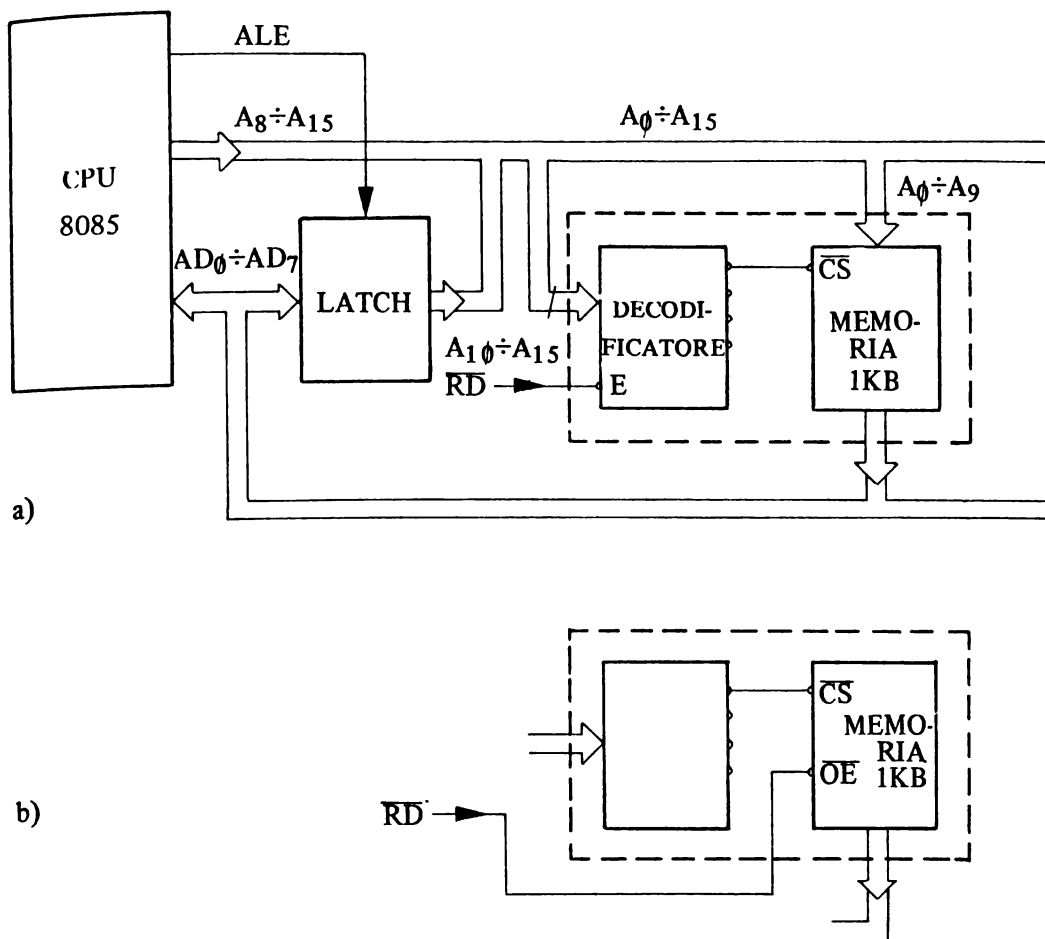



Fig. 2.55
Eliminazione del bus contention con bus dati ed indirizzi multiplexati.

I due schemi di fig. 2.55a) e b) presentano delle caratteristiche differenti per quanto riguarda l'istante in cui la memoria è in grado di fornire i dati. Nello schema di fig. 2.55a) infatti ciò avviene con un tempo di ritardo, dall'istante in cui è attivo il segnale RD/, cui contribuiscono il ritardo di propagazione del decodificatore ed il tempo t_{CO} che intercorre tra l'attivazione del segnale di abilitazione CS/ e l'istante in cui le uscite della memoria escono dallo stato di alta impedenza. Nello schema di fig. 2.55b) invece tale tempo di ritardo è quello associato all'attivazione dell'ingresso OE/, ed è di solito molto minore di t_{CO}. Ciò ha come risultato, dato che la durata del segnale RD/, fornito dal microprocessore, è limitata e determinata dal ciclo macchina che questo sta eseguendo, che per l'utilizzo dello schema di fig. 2.55a) occorrono per la memoria delle caratteristiche di velocità di risposta migliori di quelle che necessitano per la memoria della fig. 2.55b).

Capitolo 3

LE MEMORIE DINAMICHE

3.1. Generalità

In una memoria dinamica l'informazione è conservata in un condensatore che può trovarsi carico o scarico: a queste due condizioni si associano i due valori logici alto o basso. La carica nel condensatore permane solamente per un tempo limitato a causa delle inevitabili dispersioni e, quindi, se essa non è ripristinata al suo valore iniziale prima che assuma dei valori troppo piccoli, si ha la perdita dell'informazione associata a tale carica. L'operazione di ripristino della carica, detta anche rinfresco, deve avvenire, per le memorie dinamiche in commercio, entro un intervallo di tempo dell'ordine dei millisecondi ed è in genere eseguita mediante degli opportuni circuiti esterni.
Le prime memorie dinamiche realizzate presentavano in ogni cella da tre a quattro elementi attivi ed il grado di integrazione era piuttosto limitato. Attualmente le memorie dinamiche presentano delle celle costituite da un solo transistor MOS oltre al condensatore per la conservazione della informazione: questo ha permesso di raggiungere dei gradi di integrazione notevolmente più elevati così che sono ora disponibili delle memorie con 128 K celle in un singolo circuito integrato.
Ciò che ha in parte limitato il diffondersi delle memorie dinamiche è la necessità di dover disporre di un circuito per il rinfresco; per contro i principali vantaggi che esse presentano sono, data la elevata densità di integrazione raggiungibile, un basso costo e la possibilità di realizzare il sistema di memoria necessario ad un microelaboratore in poco spazio. A conferma dell'attuale interesse nell'uso di questo tipo di memorie è il fatto che sia il microprocessore Z80, sia quello più potente a 16 bit Z8000, prevedono di inviare attraverso il bus di controllo e degli indirizzi delle informazioni che facilitano l'operazione di rinfresco, semplificando quindi notevolmente la struttura hardware a ciò necessaria.
Da qualche tempo sono anche disponibili delle memorie che prendono il nome di pseudo statiche o quasi statiche: in realtà si tratta di memorie dinamiche all'interno delle quali è stato integrato anche il circuito di rin-

fresco che provvede a tale operazione in modo automatico, per cui l'utente utilizza un tale tipo di memoria praticamente con le stesse modalità di una normale memoria statica.
La tecnologia costruttiva adottata è del tipo MOS a canale p oppure, più raramente, a canale n. Sono perciò richieste di solito più tensioni di alimentazione (5V, –5V, 12V) anche se attualmente si trovano delle memorie dinamiche con una sola tensione di alimentazione di 5V. Gli ingressi e le uscite dei vari segnali sono sempre TTL compatibili per facilitare la loro utilizzazione in un normale microelaboratore.

3.2. Organizzazione interna di una memoria dinamica

Lo schema semplificato di una cella di memoria dinamica è riportato nella fig. 3.1.

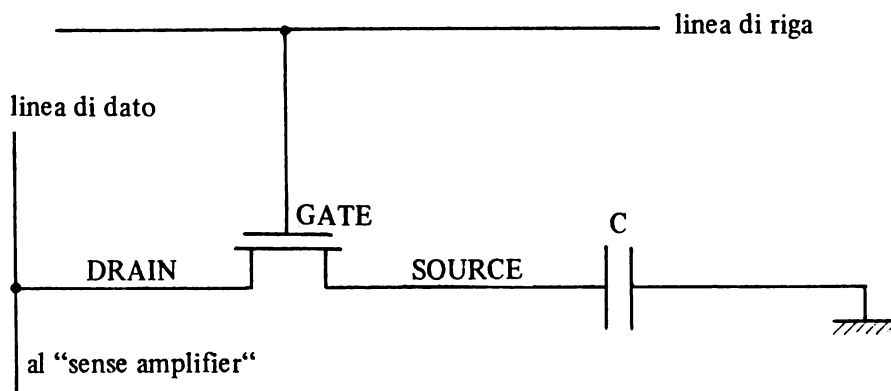



Fig. 3.1
Schema di una cella di memoria dinamica.

Come si può notare la cella è formata da un condensatore C e da un solo transistor MOS; il condensatore ha una armatura alla tensione di riferimento, mentre la tensione dell'altra armatura dipende dalla carica immagazzinata nel condensatore stesso.
Portando ad un certo valore, mediante la linea di riga, la tensione sul gate del transistor, si ottiene praticamente un collegamento diretto fra il drain ed il source, per cui la tensione della linea di dati e di una delle armature del condensatore coincidono.
Una volta selezionata la cella, inviando un valore logico alto sulla linea di riga, è allora possibile conoscere il valore della tensione ai capi del condensatore, e quindi la sua carica, analizzando la tensione che assume la linea

di dato. Attraverso tale linea è altresì possibile variare la carica del condensatore per una operazione di scrittura, oppure ripristinare il suo valore originale in una operazione di rinfresco. E' ovviamente necessaria una adatta organizzazione circuitale per poter eseguire le tre operazioni suddette: allo scopo sono predisposti degli opportuni circuiti, comandati da segnali esterni, con i quali è possibile attivare le diverse operazioni.
Le celle della memoria dinamica, come quelle di un qualunque tipo di memoria a semiconduttore, sono organizzate a matrice di m righe ed n colonne. In una memoria da 16 Kbit, ad esempio, sono presenti 128 righe e 64 colonne; l'organizzazione di una di queste colonne, in forma semplificata, è riportata nella fig. 3.2:

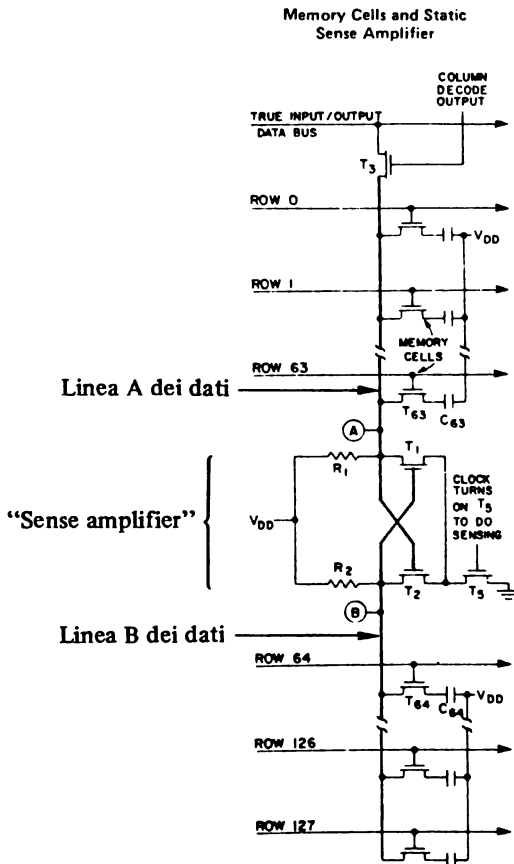



Fig. 3.2
Le celle ed il "sense amplifier" in una colonna di una memoria dinamica.
(Mostek, Memory data book and designers guide, 1980).

Si supponga di voler effettuare una operazione di lettura e di voler leggere il contenuto della cella numero 1 della colonna riportata nella fig. 3.2. La prima operazione da eseguire è di portare a livello alto la tensione della linea di riga dove si trova la cella che interessa (Row 1): in tal modo si rende conduttore il corrispondente transistor presente nella cella ed il condensatore C_1 risulta quindi collegato alla linea A dei dati. Si noti che tale linea è a sua volta connessa ad un multivibratore bistabile formato dai due transistor T_1 e T_2 e dalle rispettive resistenze R_1 ed R_2; i terminali di source dei due transistor sono collegati assieme ed al transistor T_5, il quale è reso conduttore solo quando si deve effettuare una lettura. Tale multivibratore prende il nome di "sense amplifier" ed è dimensionato in modo tale che la tensione presente sulla linea dei dati possa farlo commutare in uno dei due suoi possibili stati di funzionamento.
La tensione presente sulla linea B dei dati, poiché nessuna delle righe cui si riferisce è a livello logico alto, assume un valore, intermedio tra quelli che corrispondono al valore logico alto e a basso, valore che è imposto da una opportuna rete di polarizzazione (dummy cell) non indicata in fig. 3.2.
Per tale motivo il gate di T_1 si trova ad una tensione costante mentre il condensatore C_1, per il tramite della linea A, è in grado di fornire o di assorbire cariche dal drain del transistor T_1.
Ciò fa si che, se il condensatore era caricato ad un valore di tensione basso, il transistor T_2 si porti nello stato di interdizione e quindi T_1 in quello di conduzione; il multivibratore ha commutato in uno dei suoi due possibili stati e si deve notare che il drain di T_1 si è portato ad una tensione più bassa di quella che ha innescato il fenomeno. La commutazione opposta si ha qualora il condensatore sia inizialmente caricato ad un valore di tensione alto: T_1 si interdice e T_2 diventa conduttore. I valori finali di tensione sui drain dei transistor dipendono dal dimensionamento del multivibratore e non dalla tensione di comando fornita dal condensatore.
Mantenendo sempre allo stesso livello di tensione la linea della riga selezionata, si noti che, dopo questa prima fase, l'armatura del condensatore collegata alla linea dei dati assume il potenziale di questa, per cui si ha il ripristino della carica da esso posseduta inizialmente: si è quindi eseguita automaticamente anche l'operazione di rinfresco su quella cella di memoria.
Si deve notare anche che quando si esegue la lettura non è necessario che la carica del condensatore si trovi ai suoi valori estremi: è sufficiente che sia tale da far assumere al sense amplifier lo stato che rappresenta il bit che era immagazzinato.
A causa della struttura matriciale della memoria la selezione di una riga comporta l'attivazione di tutte le celle collegate a quella riga e quindi il rinfresco delle stesse.
In questa prima fase dell'operazione di lettura si è selezionata la riga contenente la cella, si sono rinfrescate tutte le celle appartenenti a quella riga

e si è reso disponibile il valore immagazzinato nella cella sulla linea di dato.
Per l'effettiva lettura della cella si deve ora procedere alla selezione della colonna relativa: allo scopo deve essere portato a livello alto il gate del transistor T_3, che permette quindi di mettere a disposizione sulla linea del bus di ingresso-uscita quanto è presente sulla linea di dato relativa ad una particolare colonna.
E' evidente che la linea di riga deve rimanere a livello logico alto per tutto il tempo necessario all'operazione di lettura.
Per quanto riguarda l'operazione di scrittura anch'essa si evolve in una prima fase di selezione della riga dove si trova la cella interessata. Hanno luogo ovviamente le stesse attività prima viste, per cui si ha il rinfresco delle celle appartenenti a quella riga. Successivamente il dato da memorizzare fa assumere il livello di tensione alto o basso alla linea del bus interno dei dati e si seleziona poi la colonna dove si trova la cella rendendo conduttore il transistor T_3.
In tal modo la linea dei dati relativa a quella colonna assume il valore di tensione imposto dall'esterno; poiché la linea di riga si trova ancora a livello alto, anche il condensatore corrispondente assume il valore imposto ottenendo quindi la memorizzazione del dato. Si noti che in questo caso il "sense amplifier" non è attivo in quanto non è reso conduttore il transistor T_5.
Il "sense amplifier" presente su ogni colonna, se realizzato come indicato in fig. 3.2, prende il nome di statico ed ha il notevole inconveniente di richiedere la circolazione di una corrente su una delle due resistenze R_1 o R_2 quando assume uno dei due possibili stati di funzionamento attivo. Dato che sono presenti tanti sense amplifier quante sono le colonne, ciò provoca una dissipazione di potenza che rappresenta una notevole percentuale della totale potenza richiesta per il funzionamento della memoria. Sono state recentemente realizzate delle memorie dinamiche nelle quali le due resistenze sono sostituite con componenti attivi, diminuendo così notevolmente la potenza dissipata; tali circuiti prendono il nome di sense amplifier dinamici ed il vantaggio che presentano, oltre alla riduzione della potenza consumata, è quello di un aumento della velocità di funzionamento.
Si analizza ora il comportamento dell'intero dispositivo mettendo in evidenza in particolare quali ed in quale successione debbano essere forniti dall'esterno i comandi mediante i quali si attivano le diverse operazioni.
Uno schema a blocchi tipico per tali memorie è riportato in fig. 3.3.
Si tratta di una memoria dinamica con capacità di 16348 bit organizzati in due matrici di 128 righe e 64 colonne ciascuna, per cui sono necessari 14 bit di indirizzo.
Da quanto è stato prima illustrato, si può dedurre che in una memoria dinamica i bit di indirizzo, che individuano una qualsiasi cella, non sono ne-

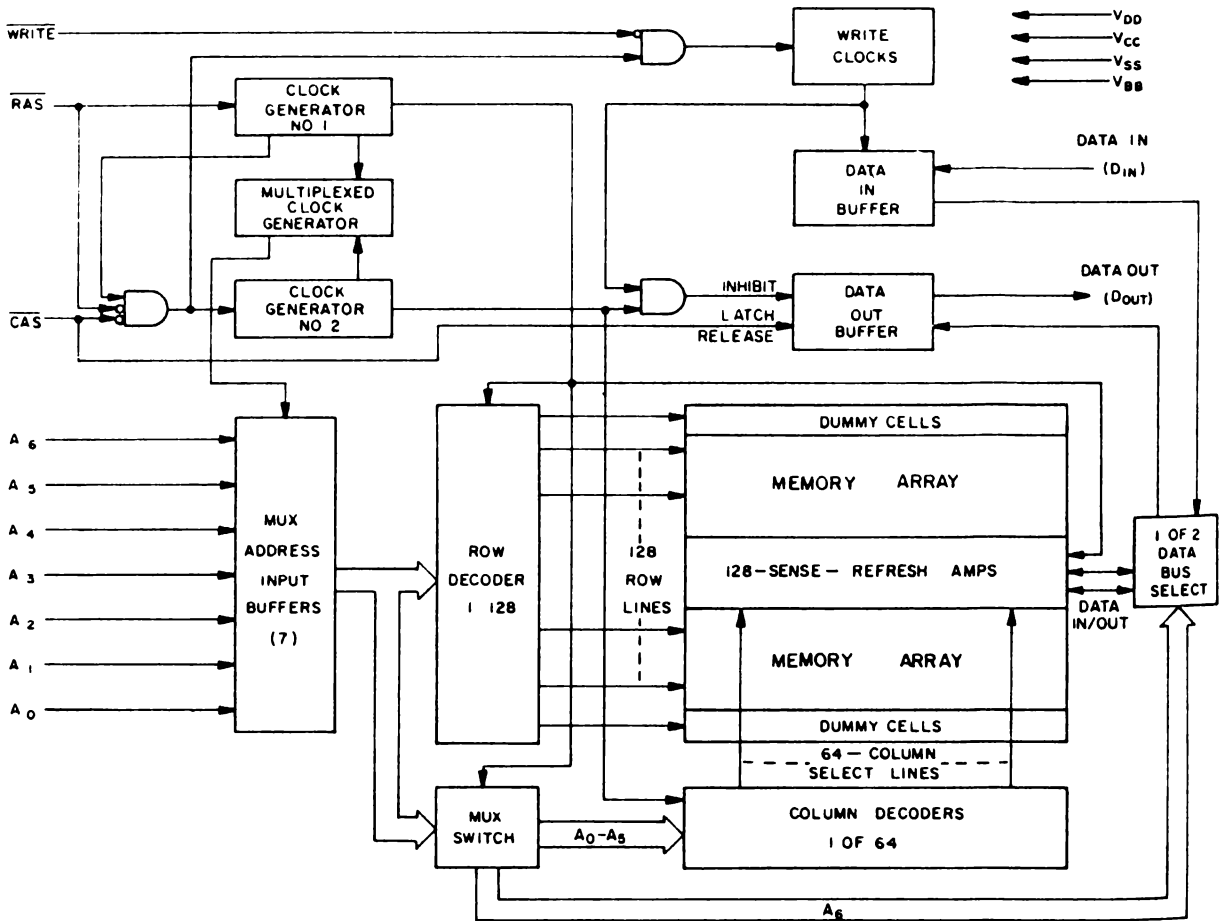
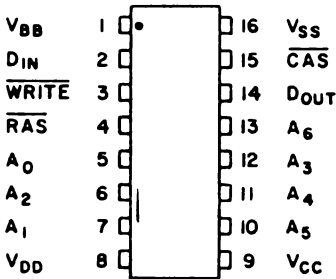



Fig. 3.3
Schema a blocchi di una memoria dinamica.
(Mostek, Memory data book and designers guide, 1980).

cessari tutti contemporaneamente: prima infatti si procede alla selezione della riga e solo in un secondo tempo si seleziona la colonna e ciò sia che si voglia eseguire una operazione di lettura che di scrittura. E' allora possibile fornire attraverso un certo numero di ingressi in un primo tempo l'indirizzo relativo alla riga, che sarà opportunamente memorizzato, e successivamente quello relativo alla colonna.
Si comprende facilmente che in tal modo sorgono alcune complicazioni, in particolare nell'hardware necessario per fornire in modo separato gli indirizzi, ma contemporaneamente si riducono il numero di piedini necessari

sul componente rendendolo più piccolo, con la possibilità di ottenere una maggior capacità di memoria su una scheda a parità di area. E' questo il principale motivo per cui è normalmente adottato tale tipo di organizzazione.
Nel caso della memoria riportata in fig. 3.3 sono allora sufficienti 7 bit per individuare una delle 128 righe, ed altri 7 bit, con 6 dei quali si individua una delle 64 colonne delle due matrici, mentre con il settimo si individua la parte di matrice cui ci si riferisce. Il blocco ": 1 of 2 data bus select" svolge appunto la funzione di mandare in attività una delle due matrici presenti. Il multiplexaggio del bus degli indirizzi permette di contenere il numero di piedini per questo tipo di memoria a 16 mentre in caso contrario ne sarebbero stati necessari almeno 22. La disposizione dei vari piedini insieme al significato di ognuno di essi è riportata nella fig. 3.4.

PIN CONNECTIONS

PIN NAMES

A_0-A_6	ADDRESS INPUTS	$\overline{WRITE}$	READ/WRITE INPUT
$\overline{CAS}$	COLUMN ADDRESS STROBE	V_{BB}	POWER (−5V)
		V_{CC}	POWER (+5V)
D_{IN}	DATA IN	V_{DD}	POWER (+12V)
D_{OUT}	DATA OUT	V_{SS}	GROUND
$\overline{RAS}$	ROW ADDRESS STROBE		

Fig. 3.4
Connessioni di una memoria dinamica.
(Mostek, Memory data book and designers guide, 1980).

Sono presenti due ingressi che permettono di specificare quanto è presente sugli ingressi relativi agli indirizzi: tali segnali sono RAS/ (row address strobe) e CAS/ (column address strobe). Con il primo si indica al dispositivo che agli ingressi di indirizzo sono presenti quelli da utilizzare per la selezione della riga, mentre con il secondo si specifica che sono presenti gli indirizzi per la selezione di una colonna.
I segnali RAS/ e CAS/ non sono utilizzati direttamente all'interno del circuito integrato ma attivano due generatori di clock con i quali si coordina il funzionamento dell'intera memoria.

Tabella 3-1: Caratteristiche temporali della memoria dinamica MK4116 (Mostek, Memory data book and designers guide, 1980).

ELECTRICAL CHARACTERISTICS AND RECOMMENDED AC OPERATING CONDITIONS (6,7,8)
($0°C < T_A < 70°C$)[1] (V_{DD} = 12.0V ± 10%; V_{CC} = 5.0V ± 10%, V_{SS} = 0V, V_{BB} = -5.7V ≤ VBB ≤ -4.5V)

PARAMETER	SYMBOL	MK 4116-2 MIN	MK 4116-2 MAX	MK 4116-3 MIN	MK 4116-3 MAX	UNITS	NOTES
Random read or write cycle time	t_{RC}	320		375		ns	9
Read-write cycle time	t_{RWC}	320		375		ns	9
Read modify write cycle time	t_{RMW}	320		405		ns	9
Page mode cycle time	t_{PC}	170		225		ns	9
Access time from $\overline{RAS}$	t_{RAC}		150		200	ns	10,12
Access time from $\overline{CAS}$	t_{CAC}		100		135	ns	11,12
Output buffer turn-off delay	t_{OFF}	0	40	0	50	ns	13
Transition time (rise and fall)	t_T	3	35	3	50	ns	8
$\overline{RAS}$ precharge time	t_{RP}	100		120		ns	
$\overline{RAS}$ pulse width	t_{RAS}	150	10,000	200	10,000	ns	
$\overline{RAS}$ hold time	t_{RSH}	100		135		ns	
$\overline{CAS}$ hold time	t_{CSH}	150		200		ns	
$\overline{CAS}$ pulse width	t_{CAS}	100	10,000	135	10,000	ns	
$\overline{RAS}$ to $\overline{CAS}$ delay time	t_{RCD}	20	50	25	65	ns	14
$\overline{CAS}$ to $\overline{RAS}$ precharge time	t_{CRP}	−20		−20		ns	
Row Address set-up time	t_{ASR}	0		0		ns	
Row Address hold time	t_{RAH}	20		25		ns	
Column Address set-up time	t_{ASC}	−10		−10		ns	
Column Address hold time	t_{CAH}	45		55		ns	
Column Address hold time referenced to $\overline{RAS}$	t_{AR}	95		120		ns	
Read command set-up time	t_{RCS}	0		0		ns	
Read command hold time	t_{RCH}	0		0		ns	
Write command hold time	t_{WCH}	45		55		ns	
Write command hold time referenced to $\overline{RAS}$	t_{WCR}	95		120		ns	
Write command pulse width	t_{WP}	45		55		ns	
Write command to $\overline{RAS}$ lead time	t_{RWL}	50		70		ns	
Write command to $\overline{CAS}$ lead time	t_{CWL}	50		70		ns	
Data-in set-up time	t_{DS}	0		0		ns	15
Data-in hold time	t_{DH}	45		55		ns	15
Data-in hold time referenced to $\overline{RAS}$	t_{DHR}	95		120		ns	
$\overline{CAS}$ precharge time (for page-mode cycle only)	t_{CP}	60		80		ns	
Refresh period	t_{REF}		2		2	ms	
$\overline{WRITE}$ command set-up time	t_{WCS}	-20		-20		ns	16
$\overline{CAS}$ to $\overline{WRITE}$ delay	t_{CWD}	60		80		ns	16
$\overline{RAS}$ to $\overline{WRITE}$ delay	t_{RWD}	110		145		ns	16

NOTES (Continued)

6. Several cycles are required after power-up before proper device operation is achieved. Any 8 cycles which perform refresh are adequate for this purpose.
7. AC measurements assume t_T = 5ns.
8. V_{IHC} (min) or V_{IH} (min) and V_{IL} (max) are reference levels for measuring timing of input signals. Also transition times are measured between V_{IHC} or V_{IH} and V_{IL}.
9. The specifications for t_{RC} (min) t_{RMW} (min) and t_{RWC} (min) are used only to indicate cycle time at which proper operation over the full temperature range (0°C ≤ T_A ≤ 70°C) is assured.
10. Assumes that t_{RCD} ≤ t_{RCD} (Max). If t_{RCD} is greater than the maximum recommended value shown in this table, t_{RAC} will increase by the amount that t_{RCD} exceeds the value shown.
11. Assumes that t_{RCD} (max).
12. Measured with a load equivalent to 2 TTL loads and 100pF.
13. t_{OFF} (max) defines the time at which the output achieves the open circuit condition and is not referenced to output voltage levels.
14. Operation within the t_{RCD} (max) limit insures that t_{RAC} (max) can be met. t_{RCD} (max) is specified as a reference point only; if t_{RCD} is greater than the specified t_{RCD} (max) limit, then access time is controlled exclusively by t_{CAC}.
15. These parameters are referenced to CAS leading edge in early write cycles and to WRITE leading edge in delayed write or read-modify-write cycles.
16. t_{WCS}, t_{CWD} and t_{RWD} are restrictive operating parameters in read write and read modify write cycles only. If t_{WCS} ≥ t_{WCS} (min), the cycle is an early write cycle and the data out pin will remain open circuit (high impedance) throughout the entire cycle. If t_{CWD} ≥ t_{CWD} (min) and t_{RWD} ≥ t_{RWD} (min), the cycle is a read-write cycle and the data out will contain data read from the selected cell. If neither of the above sets of conditions is satisfied the condition of the data out (at access time) is indeterminate.
17. Effective capacitance calculated from the equation $C = \frac{I \Delta t}{\Delta V}$ with ΔV = 3 volts and power supplies at nominal levels.
18. $\overline{CAS}$ = V_{IHC} to disable D_{OUT}.

Due buffer, uno di ingresso e l'altro di uscita, permettono il collegamento della memoria con i bus del sistema; in particolare il buffer di uscita può essere posto nello stato di alta impedenza.

3.3. Cicli di memoria

Si vogliono ora esaminare le varie relazioni temporali che debbono intercorrere fra i segnali di comando necessari per attivare le diverse operazioni della memoria.
Il significato dei vari simboli ed i relativi valori per la memoria che si sta considerando sono riportati nella tab. 3.1, assieme alle note esplicative che forniscono ulteriori precisazioni su come alcuni segnali devono essere interpretati.

3.3.1. *Ciclo di lettura*

Il diagramma temporale dei segnali che interessano durante un ciclo di lettura è riportato nella fig. 3.5. Questo ciclo inizia con l'invio agli ingressi di indirizzo dei bit che servono per individuare la riga interessata e con l'attivazione, a livello logico basso, dell'ingresso RAS/.
La prima attività intrapresa dalla memoria, essendo attivo RAS/, consiste nel convertire tale segnale da livelli TTL a quelli compatibili con la tecnologia MOS mediante il blocco "clock generator n. 1" (fig. 3.3). Quindi quanto presente sui sette piedini di ingresso di indirizzo è memorizzato ed inviato al decodificatore di riga per individuare quale delle 128 righe deve essere selezionata.
Il decodificatore, sempre su comando del generatore di clock n. 1, attiva la riga interessata per cui sulle linee dei dati all'interno della matrice è inviato quanto contenuto nelle celle di quella riga e contemporaneamente ne è assicurato il rinfresco.
Si deve procedere ora alla selezione della colonna che contiene la cella che interessa: per far ciò è necessario inviare i restanti 7 bit di indirizzo ed attivare il segnale di comando CAS/. I circuiti interni prendono in considerazione il segnale CAS/ solamente quando sono state svolte tutte le precedenti attività e non quando tale segnale viene attivato. Questo ha come conseguenza pratica che esiste una certa libertà nell'istante di attivazione di CAS/: può essere infatti attivato anche con anticipo rispetto all'istante in cui sarà effettivamente utilizzato per procedere alla fase di lettura.
Una volta che tale segnale è preso in considerazione dalla memoria, questa procede alla lettura di una delle 64 colonne che compongono ognuna delle due matrici: il dato letto è inviato in uscita dove permane disponibile solamente per un breve intervallo di tempo poiché i segnali RAS/ e CAS/ non possono essere attivi per più di 10 microsecondi.

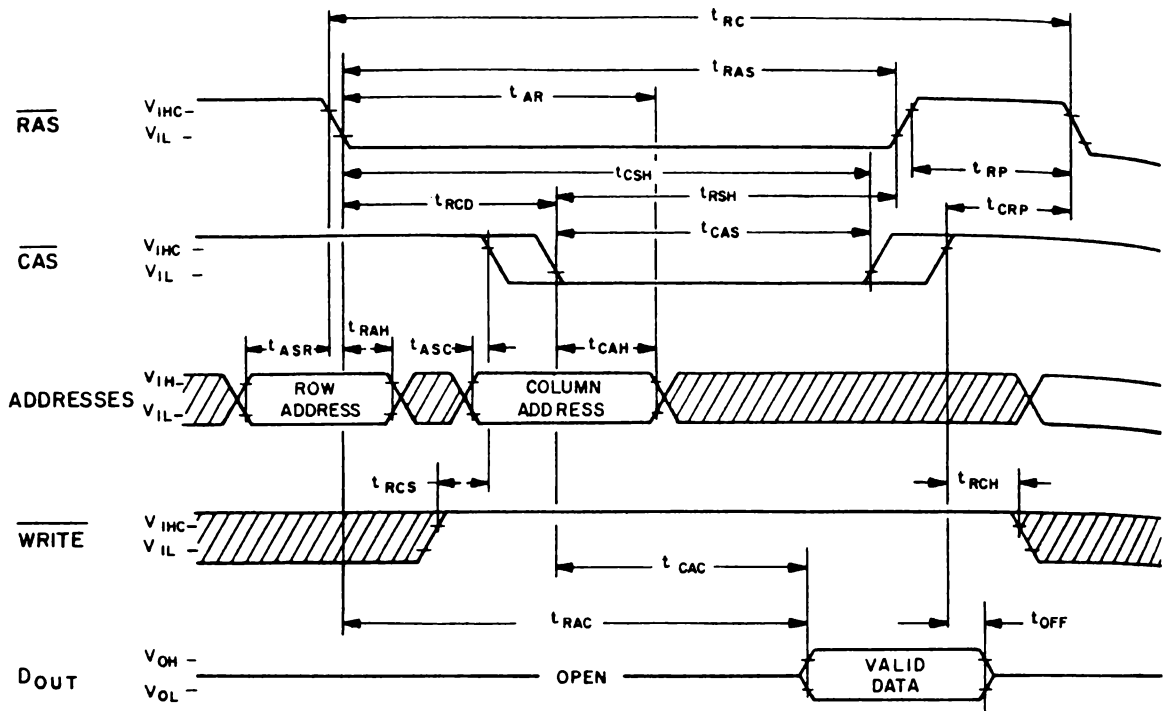



Fig. 3.5
Ciclo di lettura di una memoria dinamica.
(Mostek, memory data book and designers guide, 1980).

Per poter eseguire un'altra operazione con questa memoria è necessario che il segnale RAS/ sia disattivato e permanga in tale stato per un tempo, indicato con t_{RP}, di almeno 120 ns.

3.3.2. *Cicli di scrittura*

Si è visto che per far eseguire ad una memoria dinamica l'operazione di lettura occorre fornire in una determinata sequenza temporale i vari segnali di controllo. L'operazione di scrittura può invece avvenire con modalità diverse a seconda della sequenza con cui sono forniti i segnali di comando CAS/ e WRITE/. Anche questa operazione inizia con l'attivazione del segnale RAS/ il quale deve rimanere attivo per l'intera durata dell'operazione.

Si consideri il primo caso nel quale il segnale WRITE/ è attivato prima di CAS/; tale ciclo prende anche il nome di EARLY WRITE ed il suo diagramma temporale è riportato in fig. 3.6.

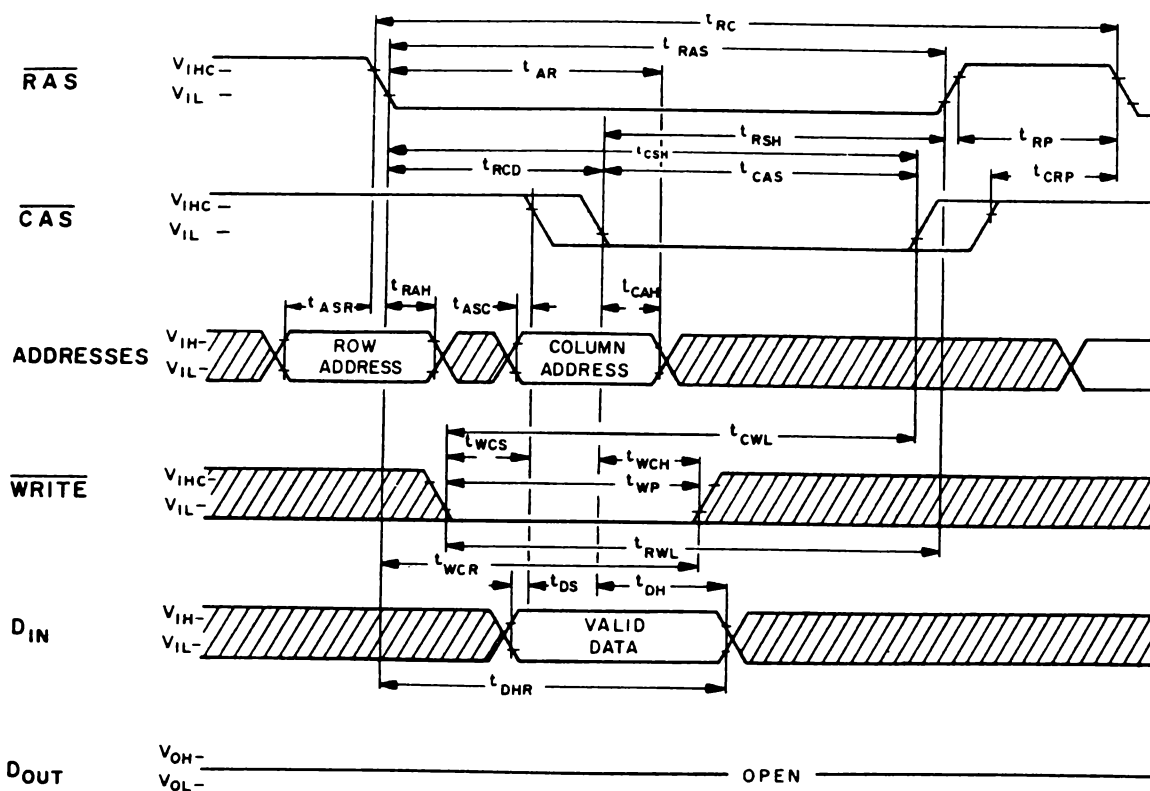



Fig. 3.6
Ciclo di scrittura di una memoria dinamica.
(Mostek, memory data book and designers guide, 1980).

Il ciclo anche in questo caso inizia inviando i segnali di indirizzo che individuano la riga dove si trova la cella interessata alla scrittura, e portando a livello logico basso l'ingresso di comando RAS/. Le attività che sono svolte in questa prima fase all'interno della memoria sono analoghe a quelle viste nel ciclo di lettura.
Successivamente deve essere attivato il segnale di WRITE/ e quindi quello di CAS/ con il quale si seleziona la colonna interessata. Il dato presente sulla linea di ingresso D_{IN} è quindi trasferito alla linea dei dati all'interno

della matrice, per cui il condensatore della cella selezionata assume la carica associata al valore del dato da memorizzare. E' importante notare che in questo caso il buffer di uscita si trova sempre nello stato di alta impedenza e quindi se si usa tale tipo di ciclo è possibile collegare fra loro le due linee D_{IN} e D_{OUT} ad un bus dati bidirezionale.
Dalla tabella 3-1 e dalla fig. 3.6 si ricava che il segnale di WRITE deve essere attivato prima di CAS/ di almeno t_{WCS} = −20 ns. Il segno meno sta a significare che in realtà WRITE/ può anche essere attivato dopo CAS/ ma con un ritardo non superiore a 20 ns; in caso contrario la memoria funziona con un ciclo diverso da quello ora descritto.
Anche in questo ciclo è necessario un periodo di inattività di RAS/ prima

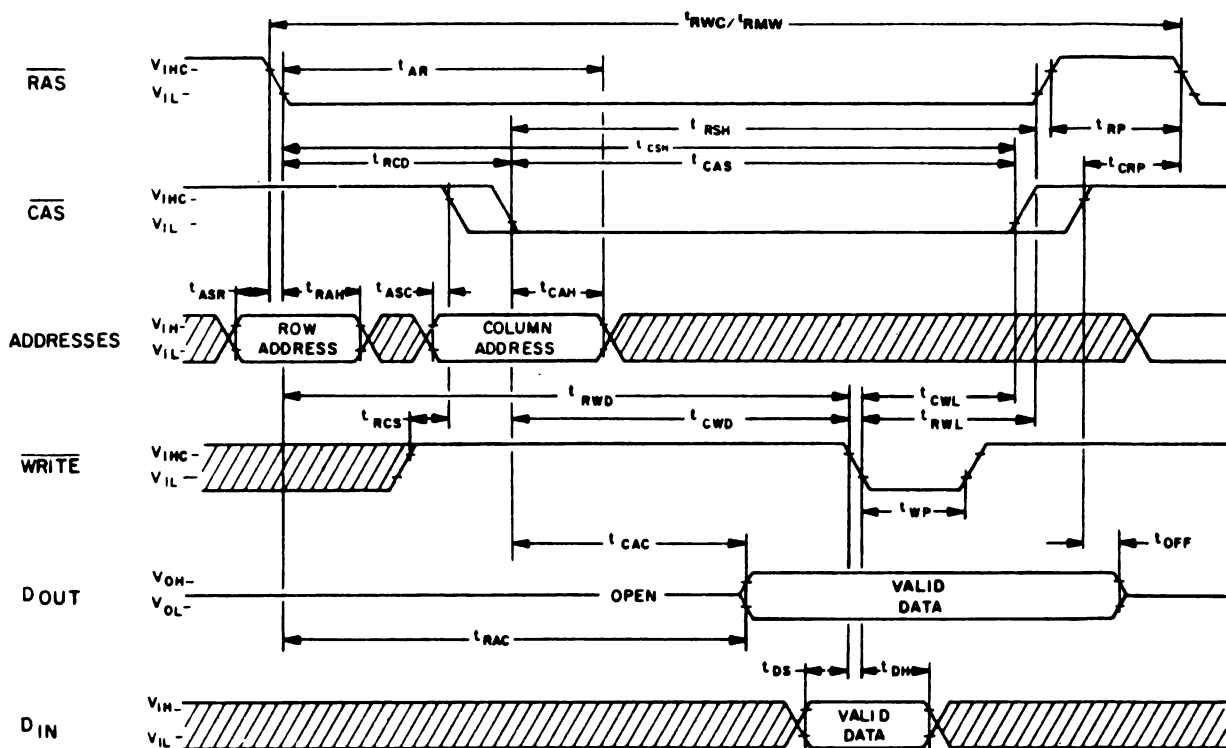
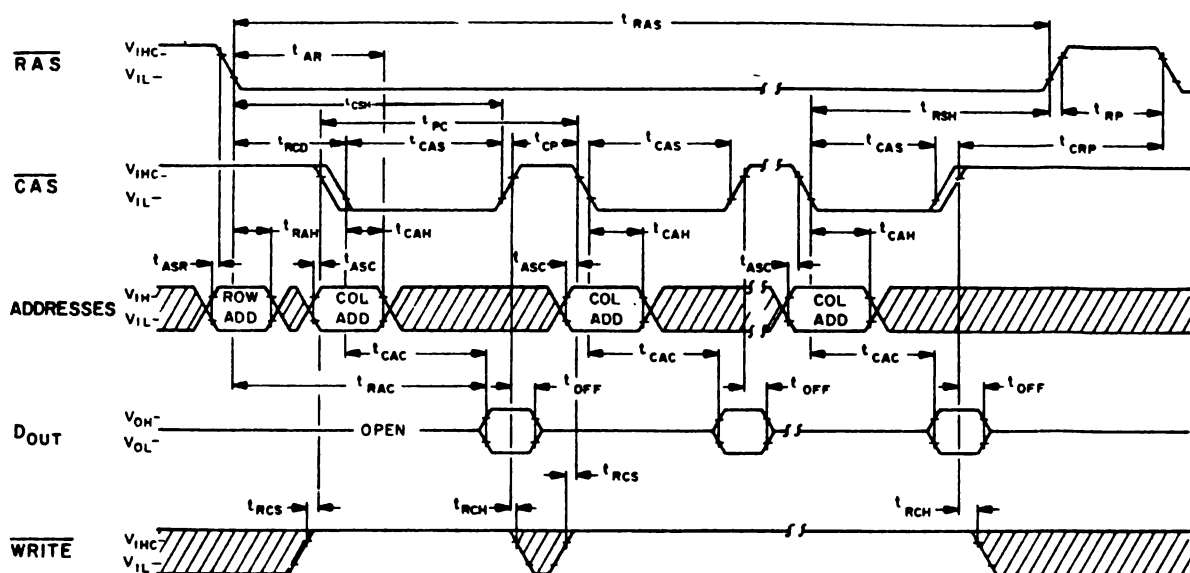



Fig. 3.7
Ciclo di lettura-scrittura e di lettura-modifica-scrittura in una memoria dinamica.
(Mostek, memory data book and designers guide, 1980).

di poter riprendere una qualsiasi operazione con la memoria. Si noti che il dato all'ingresso viene acquisito sul fronte di discesa del segnale CAS/ e deve essere ancora mantenuto stabile per un tempo: t_{DH} = 55 ns.
E' anche possibile attivare il segnale di WRITE/ dopo che è stato attivato CAS/, essendo sempre attivo RAS/. Questo ciclo prende il nome di READ-WRITE oppure di READ-MODIFY-WRITE: il relativo diagramma temporale è riportato nella fig. 3.7.
Tale ciclo inizia in modo analogo a quelli precedenti: si attivano gli indirizzi per l'individuazione della riga, il segnale RAS/ e poi viene attivato CAS/ lasciando inattivo il segnale WRITE/. In questa situazione ci si trova nelle stesse condizioni di un ciclo di lettura: è infatti possibile la selezione della cella essendo RAS/ e CAS/ entrambi attivi e poiché WRITE/ non è attivo il contenuto di tale cella viene reso disponibile all'uscita che quindi passa dallo stato di alta a quello di bassa impedenza. Solo quando è attivato il segnale WRITE/ si procede alla scrittura del dato presente all'ingresso D_{IN} sulla cella già selezionata. Si comprende quindi il motivo del nome READ-WRITE dato a tale ciclo. E' anche possibile, con un adatto hardware esterno, selezionare la cella, procedere alla lettura del dato, eventualmente eseguire delle modifiche, ed inviarne quanto risulta all'ingresso D_{IN} per scriverlo sempre nella stessa cella: si noti che non si è mai variato l'indirizzo in tutta questa successione di attività. Questo modo di usare la memoria ha dato l'altro nome a tale ciclo: READ-MODIFY-WRITE.
Nel ciclo READ-WRITE non è necessario che sia presente il dato in uscita prima di attivare il segnale di WRITE/, mentre se si utilizza il ciclo READ-MODIFY-WRITE è necessario attendere la presenza del dato letto prima di attivare il segnale WRITE/; tale segnale deve essere attivato con un ritardo, rispetto all'istante di attivazione di CAS/, almeno superiore al tempo t_{CAC}.
Per questo motivo il secondo tipo di ciclo ha una durata superiore rispetto al primo: t_{RWC} = 375 ns, t_{RMW} = 405 ns.

3.3.3. *Funzionamento a pagine*

La struttura a matrice e l'organizzazione interna delle memorie dinamiche permettono anche un altro modo di funzionamento, sia in fase di lettura che di scrittura, che prende il nome di funzionamento a pagine.
Come si è già visto, in tali memorie, una volta selezionata la riga, i dati ad essa relativi sono tutti disponibili: infatti si procede al rinfresco mantenendo semplicemente selezionata la riga. Non è allora necessario operare una nuova selezione della stessa riga per eseguire una operazione su un'altra delle celle ad essa collegate. Nell'eseguire una operazione di lettura si può procedere ad una iniziale selezione della riga che interessa e successivamente attivando e disattivando il segnale CAS/ si selezionano altre celle della

PAGE MODE READ CYCLE

PAGE MODE WRITE CYCLE

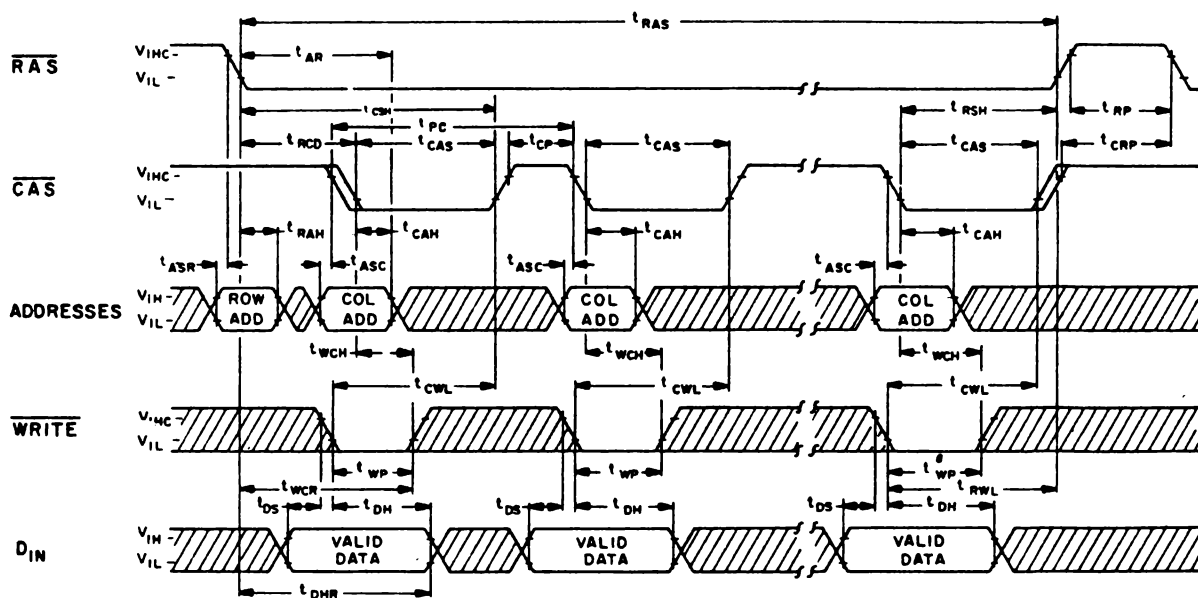



Fig. 3.8
Funzionamento a pagine di una memoria dinamica.
(Mostek, memory data book and designers data guide, 1980).

stessa riga, inviando i corrispondenti indirizzi. E' questo il funzionamento a pagina: il diagramma temporale sia per la lettura che per la scrittura è riportato nella fig. 3.8.
Come si può notare una volta selezionata la riga è sufficiente l'invio degli indirizzi della colonna e l'attivazione del segnale CAS/, oltre al segnale WRITE/ e l'invio dei dati all'ingresso se si tratta di una scrittura. Il segnale RAS/ deve rimanere attivo per tutta la durata dell'operazione a pagine. Esiste tuttavia un limite superiore che riguarda il tempo in cui RAS/ può permanere attivo: esso deve essere disattivato per un certo intervallo di tempo non inferiore a un certo valore minimo prima di poter procedere ad una nuova operazione sulla memoria. Dai diagrammi temporali e dalla tabella 3-1 risulta che al massimo RAS/ può rimanere nello stato attivo per 10.000 ns. Il periodo minimo per l'attivazione del segnale CAS/ è t_{PC} = = 225 ns: si possono perciò compiere successivamente al massimo 10.000/ /225 = 44 operazioni di lettura o di scrittura a pagina.
La possibilità di operare a pagine è molto utile quando si deve fare un trasferimento di dati tra aeree di memoria diverse, ad esempio utilizzando la tecnica del DMA.

3.3.4. *Rinfresco*

Come è già stato detto in una memoria dinamica è necessario procedere periodicamente all'operazione di rinfresco poiché in caso contrario si ha la perdita dell'informazione immagazzinata. Sia nel ciclo di lettura che di scrittura una volta selezionata la riga, è sufficiente mantenere a livello alto la linea della riga per avere automaticamente il ripristino della carica originale nei condensatori presenti su tutta la riga.
Per effettuare il rinfresco è quindi sufficiente fornire l'indirizzo della riga ed attivare il solo ingresso RAS; il diagramma temporale di tale operazione è riportato nella fig. 3.9. Ogni riga deve essere rinfrescata almeno ogni 2 ms. Un ciclo di rinfresco non può avere una durata inferiore a t_{RC} che, nel caso della memoria in esame vale 375 ns. Essendoci nella matrice 128 righe si ha a disposizione per ognuna di esse un tempo pari a 2.10^{-3}/ /128 $\cong$ 15 μs, più che sufficiente per tale operazione. Per eseguire il rinfresco di tutta la memoria è necessario avere un contatore ciclico, con modulo pari al numero di righe presenti nella memoria (che solitamente è 64 oppure 128): è quindi sufficiente un contatore con 6 o 7 bit che si incrementi automaticamente per fornire l'indirizzo delle righe da rinfrescare.

3.4. Impiego di memorie dinamiche con il microprocessore Z80

Il microprocessore Z80 agevola l'impiego di memorie dinamiche: è infatti presente al suo interno un registro a 7 bit che fornisce l'indirizzo delle ri-

"$\overline{RAS}$–ONLY" REFRESH CYCLE
NOTE: $\overline{CAS}$ = V_{IHC}, $\overline{WRITE}$ = Don't Care

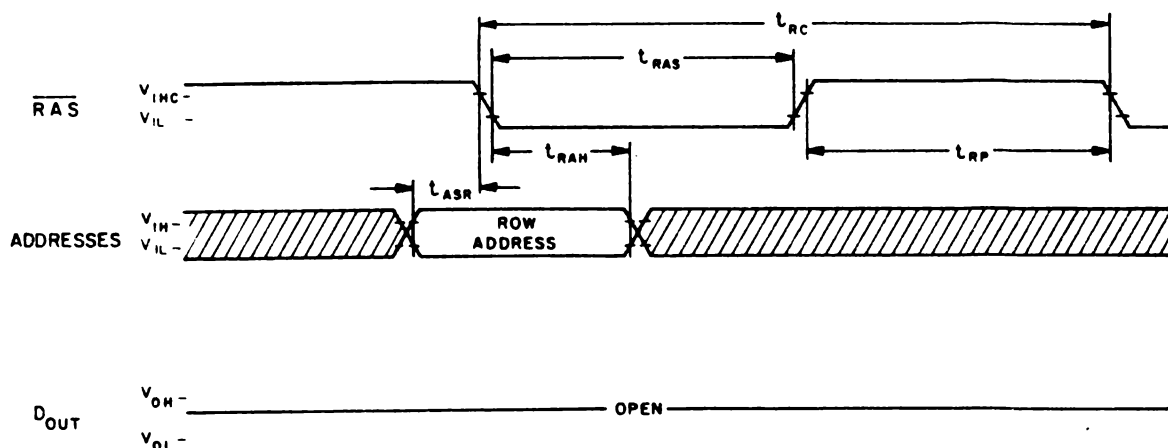



Fig. 3.9
Ciclo di rinfresco in una memoria dinamica.
(Mostek, memory data book and designers guide, 1980).

ghe per il rinfresco e che è automaticamente incrementato dopo che questo è avvenuto. Lo Z80 inoltre genera il segnale di controllo RFSH/ mediante il quale si avvisa la memoria dinamica che sul bus degli indirizzi è presente proprio l'indirizzo su cui deve essere eseguita l'operazione di rinfresco.
Questa operazione è attivata ogni volta che si esegue un ciclo di fetch di un codice operativo: non si ha quindi una cadenza fissa per il rinfresco in quanto questa dipende dal tipo di istruzioni che sono eseguite. Ci si può chiedere se tale cadenza, automaticamente imposta dall'unità centrale, è sufficiente qualunque sia il tipo di memoria dinamica utilizzata: per verificare ciò si deve considerare il massimo tempo impiegato dallo Z80 per eseguire una istruzione. Esistono delle istruzioni che abbisognano per la loro esecuzione anche di 23 periodi di clock (EX (SP),IX; SET; ecc.) per cui al massimo ogni $23 \times 400 \times 10^{-9} = 9{,}2\ \mu s$ si ha sicuramente una operazione di rinfresco e questo deve essere considerato il caso peggiore. In una memoria dinamica con 128 righe si ha a disposizione per ogni riga un tempo di circa 15 μs: si deduce che con lo Z80 è possibile, per il rinfresco, impiegare le diverse memorie dinamiche attualmente disponibili sul mercato. Ad esempio anche nel caso peggiore di istruzioni successive da 23 periodi di clock per il rinfresco della 4116-3, con 128 righe, è necessario un tempo pari a: $23 \times 400 \times 10^{-9} \times 128 = 1{,}17$ ms, inferiore a 2 ms, che è il tempo massimo di rinfresco permesso in questa memoria.

E' da tener presente che debbono essere evitate quelle situazioni nelle quali non è eseguita una operazione di rinfresco nel tempo specificato per la memoria: con la 4116 questo può avvenire se il segnale di RESET di ingresso al microprocessore è attivato per un tempo superiore ad 1 ms, oppure se si introducono degli stati di Wait consecutivi la cui durata complessiva superi 1 ms, oppure se il microprocessore cede il controllo dei bus ad altri dispositivi per un tempo dell'ordine del ms.
In tali situazioni lo Z80 non effettua il rinfresco delle memorie entro il tempo necessario e si deve perciò provvedere con un adatto circuito esterno.
Per utilizzare delle memorie dinamiche in un microelaboratore è di solito necessaria una interfaccia fra l'unità centrale e la memoria stessa. Le caratteristiche di tale interfaccia dipendono dal tipo di microprocessore utilizzato, ed in particolare dai segnali di comando presenti; devono poi essere tenute presenti le specifiche proprie delle memorie dinamiche, alle quali devono essere forniti i due segnali RAS/ e CAS/, oltre agli indirizzi di riga e di colonna, ed il segnale di comando dell'operazione da eseguire.
In fig. 3.10 sono riportati i vari segnali interessati e le relazioni temporali più importanti che devono essere prese in esame.

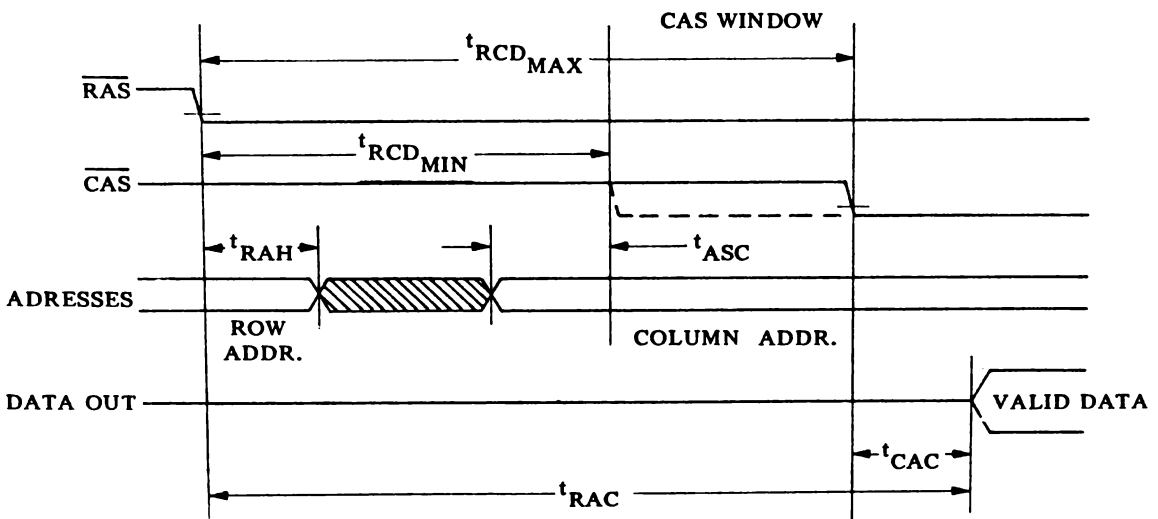



Fig. 3.10
Diagramma temporale per l'inizio di una attività in una memoria dinamica.

L'inizio di una qualsiasi operazione sulla memoria si ha portando a livello basso il segnale RAS/, nell'ipotesi che siano già stabilizzati i segnali di indirizzo di riga; questi indirizzi devono mantenersi ancora validi dopo l'at-

tivazione di RAS/ per un tempo minimo t_{RAH}. Il segnale CAS/ deve essere attivato entro un definito intervallo dall'attivazione di RAS/ ed alla sua attivazione devono essere già disponibili gli indirizzi relativi alle colonne.
I dati sono disponibili in uscita dopo un tempo t_{RAC}, tempo di accesso della memoria; esiste un altro tempo di accesso ed è quello relativo al segnale CAS/ indicato nella fig. 3.10 con t_{CAC}. Non è riportata nel diagramma temporale un'altra esigenza che deve essere soddisfatta: il segnale RAS/ deve rimanere inattivo per un certo intervallo di tempo prima di poter iniziare una nuova operazione; tale tempo è spesso indicato come tempo di precarica.

3.5. Esempio di utilizzazione di memorie dinamiche

In un microelaboratore che abbia come CPU uno Z80 si voglia realizzare un banco di memoria da 16 Kbyte impiegando le memorie dinamiche MK4116 le cui caratteristiche sono state riportate in precedenza. Si voglia inoltre allocare tale banco a partire dall'indirizzo base 4000H (fino a 7FFFH).
Un possibile schema è quello riportato nella fig. 3.11.
Sono necessari 8 circuiti integrati di memoria 4116 in quanto ognuno contiene 16384 celle di 1 bit, ed occorrono 14 bit di indirizzo per individuare una qualsiasi posizione di memoria.
Volendo realizzare una decodifica completa ed allocare il banco di memoria a partire dalla locazione 4000H, per generare il segnale RAS/, che attiva qualsiasi operazione sulla memoria, e quindi può essere considerato alla stregua del segnale CS presente nelle RAM statiche o nelle ROM, si utilizzano allora i due bit 14 e 15 del bus degli indirizzi come ingressi di un decodificatore 8205 le cui caratteristiche sono state riportate nella fig. 2.20.
Come in ogni memoria dinamica ad ingressi di indirizzi multiplexati, alla 4116 debbono essere forniti prima gli indirizzi di riga, costituiti dai primi 7 bit meno significativi del bus degli indirizzi, quindi deve essere attivato l'ingresso di controllo RAS/; successivamente debbono essere forniti i restanti 7 bit di indirizzo e poi deve essere attivato l'ingresso di controllo CAS/. Tale sequenza di operazioni è generata dal multivibratore bistabile 74S74 e dai due multiplexer 74S157: questi ultimi presentano due ingressi da quattro bit ciascuno ed una sola uscita di quattro bit; la selezione fra gli ingressi A oppure B avviene in base al livello logico assunto dall'ingresso di selezione S. La tabella della verità di tale componente è riportata in fig. 3.12.
Inizialmente il segnale MREQ/ si trova a livello logico alto, cioè disattivato, per cui il segnale SMX, in uscita dal Flip-Flop 74S74, e l'ingresso S di selezione dei 74S157 si trovano a livello basso; in tale situazione in uscita

Fig. 3.11
Banco di 16 Kbyte di memoria dinamica
con indirizzo base 4000H.

FUNCTION TABLE

INPUTS				OUTPUT Y	
STROBE	SELECT	A	B	'157, 'L157, 'LS157, 'S157	'LS158 'S158
H	X	X	X	L	H
L	L	L	X	L	H
L	L	H	X	H	L
L	H	X	L	L	H
L	H	X	H	H	L

H = high level, L = low level, X = irrelevant

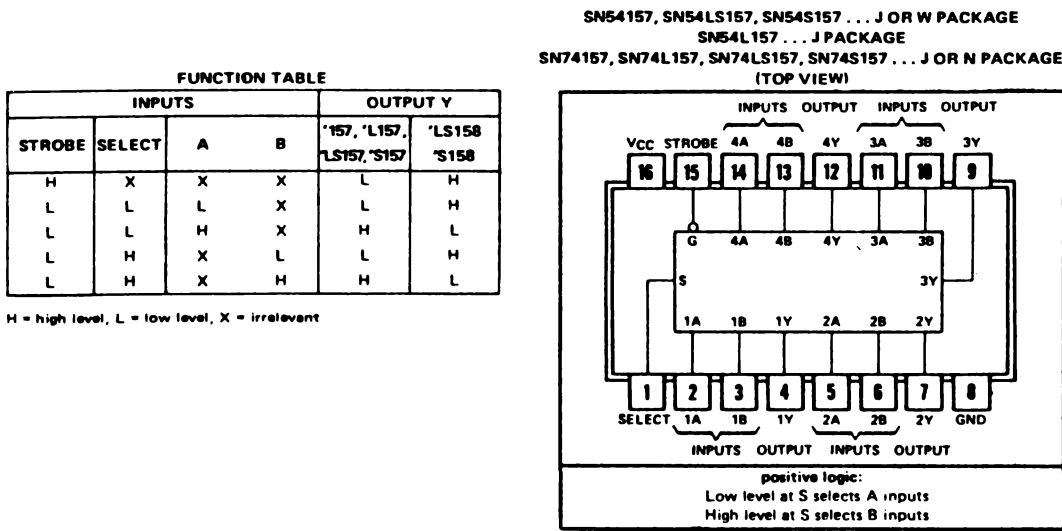



Fig. 3.12
Tabella di verità e piedinatura del multiplexer 74S157.
(Texas - Instruments. The TTL Data Book 1978)

ai multiplexer sono disponibili gli ingressi A e cioè gli indirizzi di riga, e contemporaneamente è a livello alto, cioè disattivato, CAS/.

Si supponga sia in esecuzione un ciclo macchina di fetch sul banco in oggetto per cui, dopo aver fornito gli indirizzi, il microprocessore attiva a valore logico basso il segnale MREQ/: l'uscita della porta NAND B va a livello logico basso rendendo così attivo il segnale di controllo RAS/ mediante il quale iniziano le operazioni sulle 4116.

Sul fronte di salita del successivo periodo del clock Ø il segnale SMX si porta al valore logico alto così che le uscite dei multiplexer presentano i 7 bit più significativi dell'indirizzo e contemporaneamente anche il segnale di CAS/ attivo, dato che l'ingresso 4B del multiplexer 2° è a livello logico basso.

In tal modo alla memoria sono stati forniti nella giusta sequenza tutti i comandi necessari ad una operazione di lettura.

I dati letti, forniti dalla memoria, saranno successivamente immessi nel bus dei dati quando il microprocessore attiverà il segnale di controllo RD/, il quale va ad agire sul buffer 3-state 8197.

In un ciclo di scrittura sarà invece attivato il segnale di WR/, dopo che sono state eseguite le precedenti sequenze di operazioni per l'attivazione della memoria.

In un ciclo di rinfresco saranno attivati dal microprocessore solo i segnali RFSH/ e MREQ/: il primo resetterà il bistabile 74S74 il quale farà sì, tramite i 74S157, che siano presentati alla memoria gli indirizzi relativi alle righe: quando MREQ/ diviene attivo è generato il segnale RAS/.
Da un punto di vista logico il circuito proposto può funzionare; si deve ora verificare se sono soddisfatte le relazioni temporali di specifica fra i diversi segnali per un corretto funzionamento.
Si consideri il diagramma temporale della fig. 3.13 dove sono riportati i segnali che ora interessano, e si prenda come riferimento l'istante in cui viene attivato il segnale MREQ/: tale segnale diventa attivo al massimo 100 ns dopo il fronte di discesa del clock nel periodo T_1; l'attivazione del segnale RAS/ avviene con un ritardo pari alla somma dei ritardi introdotti dalle due porte 7404 e 7400 pari al massimo a 22 + 15 = 37 ns.

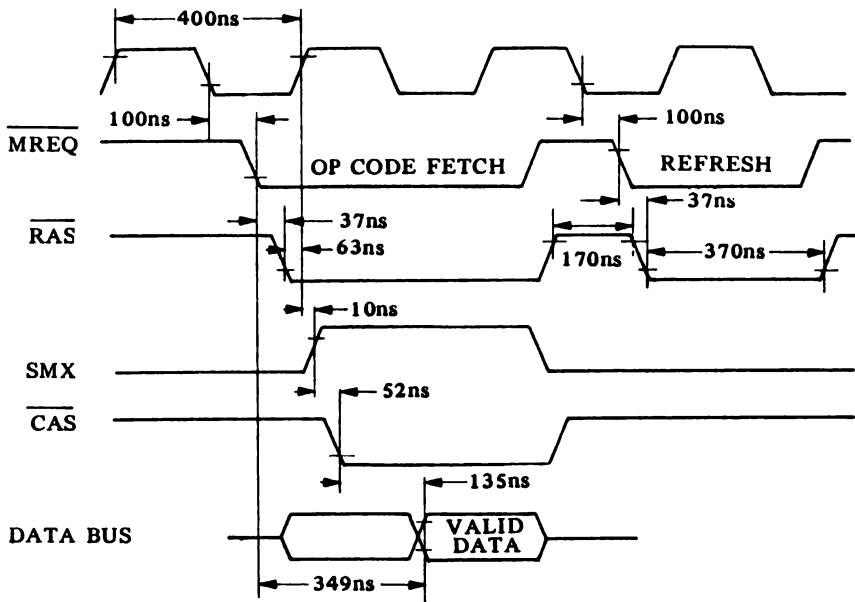



Fig. 3.13
Diagramma temporale relativo al banco di memoria di fig. 3.11.

Quindi RAS/ è attivato con un anticipo di: 200 − 100 − 37 = 63 ns rispetto al fronte di salita del periodo di clock T_2, in cui si attiva il segnale SMX, con un ritardo, dovuto al FF, pari a 10 ns.
Perché il segnale CAS/ sia attivo si deve aspettare il ritardo introdotto dal multiplexer 74S157, pari a 15 ns, e quello, introdotto dalle due porte 7404 in serie, che vale 15 + 22 = 37 ns: in totale si ha quindi un ritardo di 37 + 15 = 52 ns.

Dall'istante di attivazione del segnale CAS/ è necessario aspettare un tempo t_{CAC} = 135 ns al massimo per avere a disposizione il dato richiesto sulle linee di uscita delle memorie. Perché il dato sia disponibile sul bus dei dati si deve aggiungere il ritardo introdotto dal buffer 8197 che risulta di 22 ns.
Da quando viene attivato il segnale MREQ/ a quando il dato è disponibile sul data bus è necessario un intervallo di tempo pari a: 37 + 63 + 10 + + 52 + 135 + 22 = 319 ns nella peggiore delle situazioni. Si era visto che con lo Z80 per una corretta acquisizione del dato richiesto è necessario che questo sia disponibile al massimo dopo 450 ns dall'attivazione di MREQ/ per cui il tipo di memoria scelto ed il circuito adottato permettono un funzionamento in fase di lettura senza la necessità di introduzione di stati di Wait. Si noti che in questa analisi non sono tenuti presenti eventuali buffer presenti nel collegamento della CPU al bus, buffer che introducono ulteriori ritardi. Anche il tempo di disattivazione del segnale RAS/ è sufficiente in quanto uguale a quello in cui MREQ/ rimane disattivato, pari a 170 ns, come calcolato in precedenza.
Nello schema indicato nella fig. 3.11 è possibile l'allocazione del banco di memoria solamente a partire dall'indirizzo 4000H. Con una adatta organizzazione è anche possibile ottenere, mediante intervento manuale, la variazione dell'indirizzo base della memoria.
Le precedenti analisi temporali hanno significato in quanto si sono prese in considerazione le situazioni peggiori che si possono verificare nei confronti della memoria. Se in effetti tale memoria non serve nei cicli di fetch, avendo più tempo a disposizione, si potrebbe utilizzare anche una memoria più lenta, cioè con un tempo di accesso più lungo, e quindi di costo minore.

3.5.1. *Circuiti per la generazione dei segnali di comando RAS e CAS*

Nei circuiti esaminati è stato utilizzato il segnale di clock dell'unità centrale per una corretta sincronizzazione delle memorie dinamiche.
Uno schema a blocchi di un circuito che non utilizza tale segnale è riportato in fig. 3.14.
Il segnale RAS/ è generato dall'attivazione di MREQ/, quando il bus degli indirizzi è stabilizzato e il multiplexer è già predisposto a selezionare gli indirizzi di riga. Con un certo ritardo il multiplexer selezionerà gli indirizzi di colonna, dopo di che si attiva anche CAS/.
Il microprocessore Z80 può anche funzionare con una frequenza di clock di 4 MHz (il componente prende il nome di Z80A): se si calcola in tale caso il tempo di accesso a disposizione di una memoria interessata ad un ciclo di fetch si trova che esso è al massimo di 265 ns.
La memoria utilizzata, come già visto, ha un tempo di accesso, T_{RAC}, di

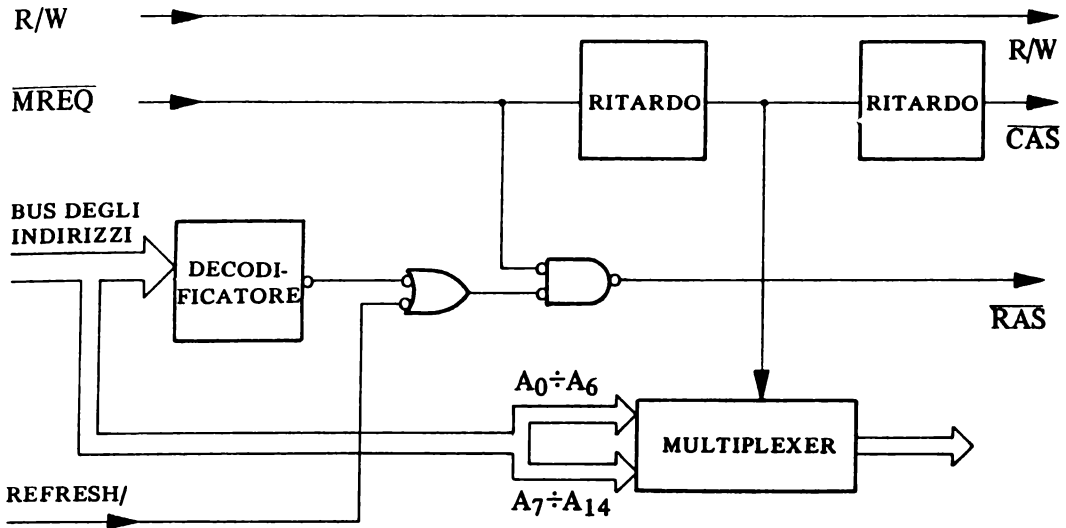



Fig. 3.14
Circuito per la generazione di segnali di controllo per memoria dinamiche.

200 ns, per cui si deve verificare se può essere ancora utilizzata, tenendo conto dei vari ritardi introdotti dagli elementi circuitali che costituiscono l'interfaccia.
Si può anche utilizzare una versione più veloce della stessa memoria (MK4116-2) che presenta un tempo di accesso di 150 ns.
Si ricordi ancora che oltre ad avere una sufficiente velocità di funzionamento la memoria richiede anche che dopo ogni operazione ci sia un intervallo di inattività o di precarica, la cui durata nella versione più veloce (4116-2) è di almeno 100 ns, pari proprio al tempo in cui il segnale MREQ/ rimane a livello alto. Per sicurezza conviene allungare tale tempo per permettere un corretto funzionamento del componente: si può allo scopo utilizzare il circuito, riportato in fig. 3.15, il cui diagramma temporale è in fig. 3.16.
Come si può notare il tempo in cui il comando RAS/ rimane attivo è sicuramente superiore a t_{RAS} = 150 ns richiesto dalla memoria; si ottiene contemporaneamente un allungamento del tempo in cui il segnale RAS/ rimane inattivo (126 ns) e cioè rispetta quindi le relazioni temporali richieste per un corretto funzionamento.

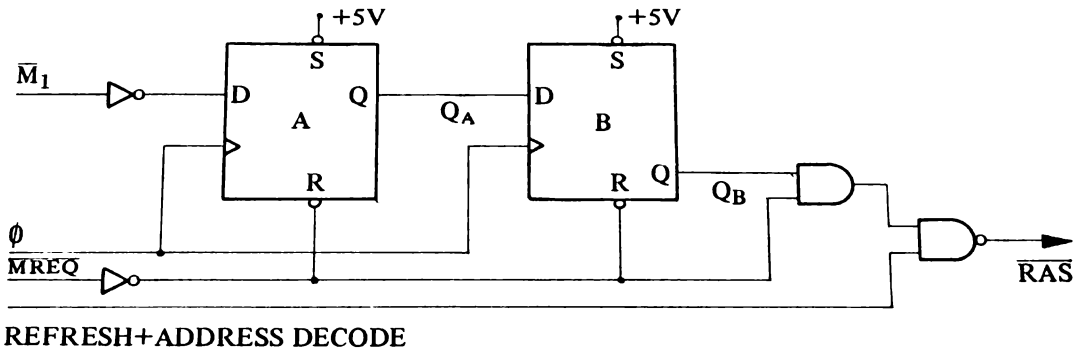



Fig. 3.15
Generazione del comando RAS/ con microprocessore Z80A (clock = 4 MHz).

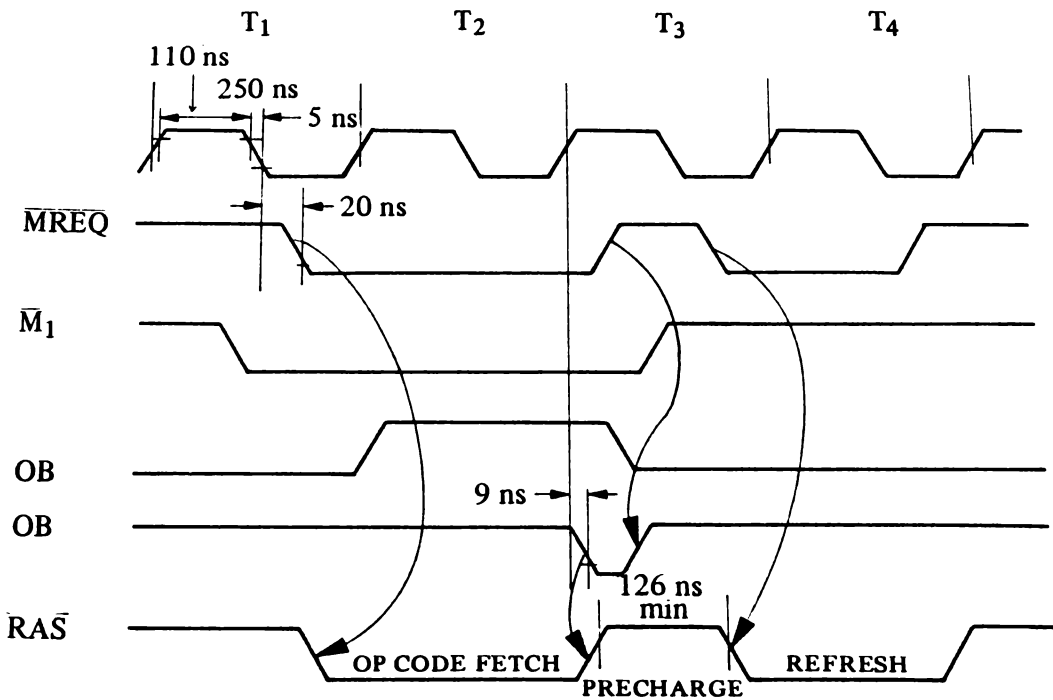



Fig. 3.16
Diagramma temporale relativo ai segnali del circuito di fig. 3.15.

3.6. Memorie pseudostatiche

Le memorie pseudostatiche sono delle memorie dinamiche al cui interno è realizzato un circuito che rende automatica l'operazione di rinfresco, per cui le operazioni di lettura e di scrittura non differiscono da quelle di una normale memoria statica.
Oltre ai vari ingressi di controllo CS/, RD/, WR/, e OE/ queste memorie ne presentano uno che, se attivato, fa eseguire l'operazione di rinfresco, la quale avviene automaticamente anche dopo ogni operazione di lettura o di scrittura.
In genere esse contengono nel loro interno un contatore binario che genera in modo automatico gli indirizzi di riga ed i segnali di RAS/ e CAS/, senza interferire durante le operazioni di lettura o di scrittura. L'ingresso per attivare il rinfresco è collegabile direttamente ad un segnale di controllo opportuno, come potrebbe essere quello generato da uno Z80, oppure fornito da un semplice circuito che lo genera durante opportuni periodi del ciclo istruzione, in cui si è certi che non saranno eseguite operazioni di lettura o di scrittura riguardanti la memoria.

Capitolo 4
TECNICHE DI INTERRUZIONE NEI MICROELABORATORI

4.1. Generalità

In un microelaboratore lo scambio di informazioni interessa l'unità centrale e la memoria, oppure la prima e gli organi di ingresso e uscita.
Lo scambio con la memoria procede con le modalità viste nei cap. 2 e 3: è sempre il microprocessore che impone come e quando deve avvenire il trasferimento; esso infatti fornisce tutti i segnali necessari alla memoria, la quale, sia nel ruolo di ricevitore che in quello di trasmettitore, esegue il trasferimento in intervalli di tempo determinati.
Nel caso invece in cui il trasferimento interessi organi di ingresso o di uscita si possono presentare delle situazioni che impongono di adottare soluzioni diverse a seconda dei dispositivi periferici interessati e delle prestazioni che si desiderano ottenere.
Infatti mentre una memoria è sempre disponibile, una volta attivata, ad accettare o ricevere un dato, ciò può non succedere per un dispositivo di ingresso o di uscita: una tastiera non può inviare un dato se non è stato premuto un tasto; una telescrivente non può accettare un nuovo dato finché non ha terminato la stampa del precedente.
In questi casi è necessario che il microprocessore, prima di iniziare un qualsiasi trasferimento, si assicuri che il dispositivo periferico con cui desidera comunicare sia in grado di partecipare correttamente al colloquio.
Si può organizzare la comunicazione con due tecniche differenti: quella a scansione (polling) o quella con richieste di interruzione.

4.2. Polling

Affinché si possa adottare questa tecnica è necessario che ogni dispositivo periferico sia dotato di uno o più indicatori di stato (flags) mediante i quali esso segnala le diverse situazioni in cui può trovarsi: occupato, libero, pronto ad accettare o ad inviare un dato, ecc.

Tali indicatori sono in genere realizzati con un registro che deve presentare delle caratteristiche simili a quelle di una locazione di memoria: è sempre possibile il prelievo dell'informazione in esso contenuta, da parte dell'unità centrale, con modalità uguali a quelle viste per la memoria.
Nella tecnica del polling il microprocessore procede ciclicamente, e secondo una sequenza ordinata, alla lettura dello stato delle varie periferiche ed in base ad esso intraprende determinate azioni.
Per una tastiera, ad esempio, il microprocessore effettua l'acquisizione del dato solo se c'è l'indicazione di una avvenuta attivazione di un tasto; in caso contrario esso prosegue nell'analisi ciclica degli stati delle eventuali altre periferiche presenti nel sistema. In fig. 4.1 è indicata in forma simbolica la tecnica di polling.

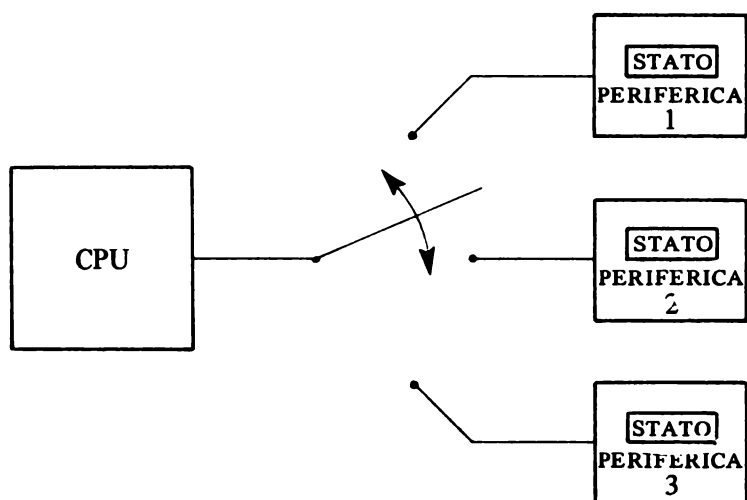



Fig. 4.1
Tecnica di polling.

Essa presenta notevoli vantaggi: è infatti semplice da realizzare, in quanto richiede una semplice struttura hardware, costituita dal registro di stato; è anche facile da gestire per quanto riguarda il software. Il suo principale inconveniente deriva dal fatto che occorre tenere impegnata l'unità centrale al solo fine di controllare lo stato delle periferiche, con un limitato sfruttamento della potenzialità del processore.
Si pensi come esempio ad un microelaboratore dotato di una unica periferica, una stampante, la cui velocità massima di stampa sia di 10 caratteri al secondo: una volta ricevuto un carattere da stampare, tale periferica inizia

la sua attività, indicando contemporaneamente di essere occupata, ed in tale stato permane al minimo per 100 millisecondi. Il microprocessore per inviare il dato impiega un tempo dell'ordine della decina di μs; se ne deve inviare un altro esso deve analizzare continuamente lo stato della stampante e solo dopo 100 ms = 10 x 10.000 μs si accorge che ciò è possibile: si è quindi nella situazione in cui il microprocessore impiega 9.990 μs per eseguire il controllo dello stato della periferica e solo 10 μs per inviare il dato.
Si possono avere delle situazioni nelle quali non è possibile adottare la tecnica del polling: infatti se ci sono più periferiche non è più fisso l'intervallo di tempo che intercorre tra due successive analisi della disponibilità di una stessa periferica; tale intervallo dipende infatti dallo stato in cui si trovano le altre e dal tempo impiegato per gestire quelle che sono disponibili al trasferimento di informazioni. Ad esempio in una trasmissione seriale sincrona non è possibile adottare tale tecnica in quanto, se il microprocessore non invia il dato in tempo utile, la periferica cui è destinato riceve una informazione errata.
Esistono anche altre situazioni, come ad esempio una segnalazione di un allarme per il verificarsi di una anomalia in una macchina controllata da un microprocessore, nelle quali non è accettabile il ritardo con il quale, se si usa la tecnica del polling, si interviene sulla macchina.
Per ovviare a tali inconvenienti si deve ricorrere alla tecnica dell'interruzione.

4.3. Interruzione

Nel polling il microprocessore deve leggere lo stato di una periferica prima di intraprendere l'azione che la riguarda: con la tecnica dell'interruzione è invece la periferica che invia, su un ingresso a ciò predisposto del microprocessore, una segnalazione per richiedere l'inizio immediato di una determinata azione. L'attivazione di tale ingresso fa si che il microprocessore interrompa l'esecuzione del segmento di programma in cui è impegnato, salvo riprenderla dopo aver eseguito la routine di servizio richiesta dalla periferica interrompente: ciò è indicato in forma schematica in fig. 4.2.
In tal modo il tempo dedicato dal microprocessore per gestire la periferica risulta molto ridotto: nell'esempio precedente se la stampante invia una richiesta di interruzione solo nel momento in cui essa si rende libera, il microprocessore può impiegare per l'esecuzione di altri programmi quasi completamente i 100 millisecondi che, con la tecnica del polling, esso sprecava al solo scopo di verificare quando era possibile l'invio di un nuovo dato.

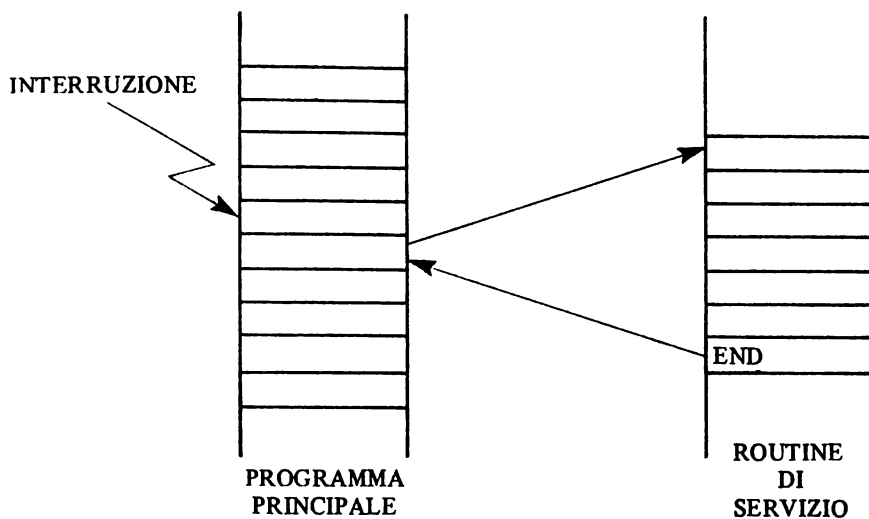



Fig. 4.2
Interruzione del programma principale ed esecuzione di una routine di interruzione.

4.4. Caratteristiche del segnale di interruzione

La periferica che fa una richiesta di interruzione ha di solito un funzionamento asincrono con quello dell'unità centrale per cui la richiesta può avvenire in qualsiasi istante durante la fase di fetch o l'esecuzione di una istruzione da parte del microprocessore.
Si è prima detto che, se esaudita, la richiesta provoca l'esecuzione di una routine di servizio: evidentemente è necessario che, eseguita quest'ultima, la CPU riprenda l'esecuzione del programma interrotto nelle stesse identiche condizioni in cui essa si trovava quando era passata a servire la richiesta di interruzione. La CPU si trova in una condizione ben determinata solamente alla fine dell'esecuzione di ogni singola istruzione: infatti solo in quel momento si stabilizzano e sono determinati i valori dei vari flags di stato, i contenuti dei registri, ecc.
La richiesta di interruzione sarà allora presa in esame solo alla fine della esecuzione dell'istruzione in corso: è quindi necessario che il segnale di interruzione permanga attivo fino a quando la CPU lo può prendere in considerazione. Occorre perciò un dispositivo che memorizzi l'avvenuta richiesta di interruzione per tutto il tempo che è necessario: questo elemento può essere interno al microprocessore stesso, oppure deve essere predisposto all'esterno.
Nei microprocessori il numero di ingressi per la richiesta di interruzioni è limitato: ad esempio l'8080 ne ha solo uno, lo Z80 due e l'8085 ne presenta cinque; tali ingressi possono inoltre presentare delle caratteristiche di attivazione molto diverse.

Qualora si presenti la necessità di inviare più richieste sull'unico ingresso presente si deve ricorrere ad un adatto hardware esterno, con una realizzazione pratica diversa a seconda della logica utilizzata. In fig. 4.3 è indicato come si possono far pervenire ad un solo ingresso di un microprocessore, attivo a livello logico basso, più richieste di interruzioni attive a livello logico alto.

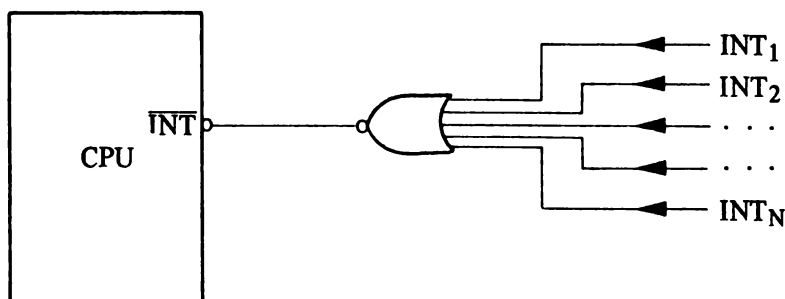



Fig. 4.3
Invio di più richieste di interruzione.

In questo caso sorge il problema di distinguere la sorgente che ha fatto la richiesta: sono possibili allo scopo varie soluzioni che saranno analizzate fra breve.

4.5. Risposta ad una richiesta di interruzione

Alla fine dell'esecuzione dell'istruzione in progresso, se sono soddisfatte determinate condizioni, il microprocessore manda in esecuzione il programma di servizio relativo alla richiesta di interruzione.
Per ottenere ciò esistono modalità diverse, che dipendono dal tipo di microprocessore, e che si possono suddividere in due categorie fondamentali: nella prima il program counter è caricato con un indirizzo prefissato, da cui inizia il programma di servizio, nella seconda esso è invece caricato con un indirizzo che è dedotto da informazioni fornite dallo stesso dispositivo interrompente.
Il primo modo di operare richiede una struttura hardware esterna estremamente semplice, ma risulta più gravoso da un punto di vista software, se come spesso succede, si deve individuare quale delle periferiche richiede il servizio.
Nel secondo modo tale informazione deve essere fornita direttamente dalle periferiche, su richiesta del microprocessore, tramite una struttura hardware aggiuntiva.

La richiesta di questa informazione da parte del microprocessore è segnalata con l'attivazione di un segnale di controllo, generalmente chiamato Interrupt Acknowledge (INTA). In fig. 4.4 è indicato lo schema a blocchi della struttura che qui interessa.

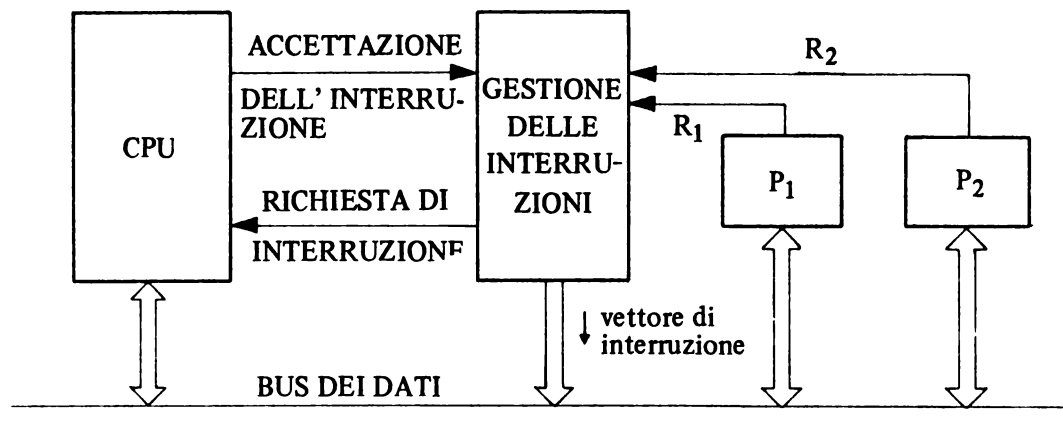



Fig. 4.4
Schema a blocchi dell'hardware di gestione delle interruzioni.

Come si vede il circuito di gestione delle interruzioni, oltre ad effettuare la richiesta della interruzione, si incarica anche di fornire al microprocessore un'ulteriore informazione tramite il bus dei dati.
La struttura hardware del blocco di gestione delle interruzioni può risultare anche piuttosto complessa: devono infatti essere rispettati dei legami temporali dipendenti dal microprocessore, ecc. Sono attualmente reperibili dei circuiti integrati che forniscono le prestazioni richieste, ed in alcuni casi il circuito è integrato nelle porte di I/O, come ad esempio per i componenti periferici che appartengono alla famiglia del microprocessore Z80; ciò semplifica notevolmente la realizzazione di un microelaboratore.
Il comportamento del microprocessore per dare inizio al servizio richiesto è analogo alla esecuzione di una istruzione di CALL: nel primo modo di gestione l'indirizzo necessario per l'esecuzione di tale istruzione è prefissato; nel secondo esso deve essere fornito dall'esterno con modalità diversa: potrebbe essere composto di 8 o 16 bit o addirittura potrebbe consistere nel codice operativo di una particolare istruzione di CALL.

4.6. Salvataggio dello stato della macchina

Per una corretta gestione delle interruzioni è necessario, come è stato detto, che, una volta terminato il programma di servizio, il microprocessore riprenda l'esecuzione del programma interrotto: è allora indispensabile siano salvate tutte quelle informazioni che ne permettono una corretta ripresa. Tutti i microprocessori indistintamente procedono al salvataggio del program counter, il quale contiene l'indirizzo dell'istruzione successiva a quella in cui è avvenuta la richiesta di interruzione; è questa l'informazione minima necessaria per poter riprendere correttamente l'esecuzione del programma interrotto.
Devono inoltre essere salvate tutte quelle informazioni, suscettibili di essere alterate dall'esecuzione della routine di servizio, contenute nei vari registri, il cui contenuto costituisce lo stato del microprocessore. In qualche caso è il microprocessore stesso, come accade per il 6800 e per lo Z8000, che provvede al salvataggio completo dello stato; in altri invece si deve provvedere, con un software opportuno, all'inizio della routine di servizio.
Per alcuni tipi di microprocessori come ad esempio lo Z80, l'intero gruppo di registri e flags è duplicato e con una sola istruzione è possibile passare da un gruppo all'altro. Con questo sistema il salvataggio dello stato si attua agevolmente: si noti tuttavia che il banco alternativo di registri può essere utilizzato per salvare lo stato precedente ad una prima interruzione, e se durante l'esecuzione della relativa routine è attivata un'altra richiesta, si deve salvare lo stato attuale via software.
Invece nel processore 9900, della Texas, tutti i registri disponibili sono allocati nella memoria, cioè esternamente alla CPU, mentre all'interno del microprocessore è presente un registro puntatore che contiene l'indirizzo in cui inizia tale area. In questo caso il salvataggio dello stato richiede semplicemente di salvare il contenuto del puntatore e di fissare un'altra area di lavoro.
Nella maggior parte dei microprocessori il salvataggio dello stato della CPU si attua, via software, in una particolare area di memoria RAM organizzata a pila, detta stack. Nell'ambito di tale area una locazione è individuata dal contenuto di un registro puntatore, detto stack-pointer, il quale è automaticamente aggiornato ogni volta che, tramite opportune istruzioni, si effettua una operazione di scrittura o di lettura su detta cella.
Lo stack di memoria si comporta come una pila in cui la prima informazione memorizzata è anche l'ultima che si può estrarre: in questo modo esso consente una facile gestione delle interruzioni a più livelli.

4.7. Richieste multiple

Nel caso siano presenti in un sistema più dispositivi in grado di effettuare richieste di interruzioni, è necessario che il microprocessore individui quale di essi abbisogna di servizio per poter mandare in esecuzione la routine appropriata.
Si possono presentare varie situazioni: nei microprocessori che presentano più di un ingresso per la richiesta di interruzione, se ad ognuno di essi è collegata una sola periferica, la individuazione avviene in maniera automatica e non è richiesto nessun hardware aggiuntivo. Quando invece più dispositivi sono collegati al medesimo ingresso di richiesta di interruzione, l'identificazione può avvenire con due tecniche differenti: quella del polling e quella della vettorizzazione.
La prima tecnica è simile a quanto già visto precedentemente: ogni dispositivo ha la sua linea di richiesta collegata, oltre all'unico ingresso di interruzione del microprocessore tramite una porta OR in comune con gli altri, anche ad una linea di una porta di ingresso. Il microprocessore, dopo aver accettato la richiesta, e quindi iniziato la routine di servizio associata a quell'ingresso di interruzione, per prima cosa acquisisce quanto presente alla porta di ingresso: ciò gli permette di individuare quali delle periferiche abbisognano del servizio e quindi può procedere all'esecuzione delle routine relative.
Uno schema a blocchi del polling nelle interruzioni è indicato in fig. 4.5.

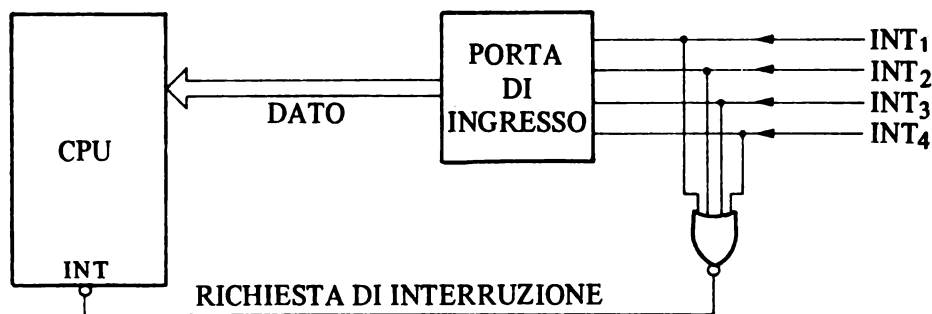



Fig. 4.5
Tecnica di polling sulle interruzioni.

Il notevole vantaggio di un sistema di interruzione a polling rispetto al polling normale prima esaminato è che quando il microprocessore esegue la scansione dei vari ingressi della porta sicuramente ne troverà almeno uno attivo.

Nella tecnica di vettorizzazione invece ciascuna sorgente di interruzione fornisce direttamente alla CPU un dato (vettore) che serve per generare l'indirizzo di inizio della routine di servizio che le compete.
La tecnica della vettorizzazione fa si che la risposta della CPU alla richiesta sia più rapida rispetto a quella del polling sulle interruzioni, ma contemporaneamente richiede dell'hardware aggiuntivo: uno schema a blocchi è riportato nella fig. 4.6.

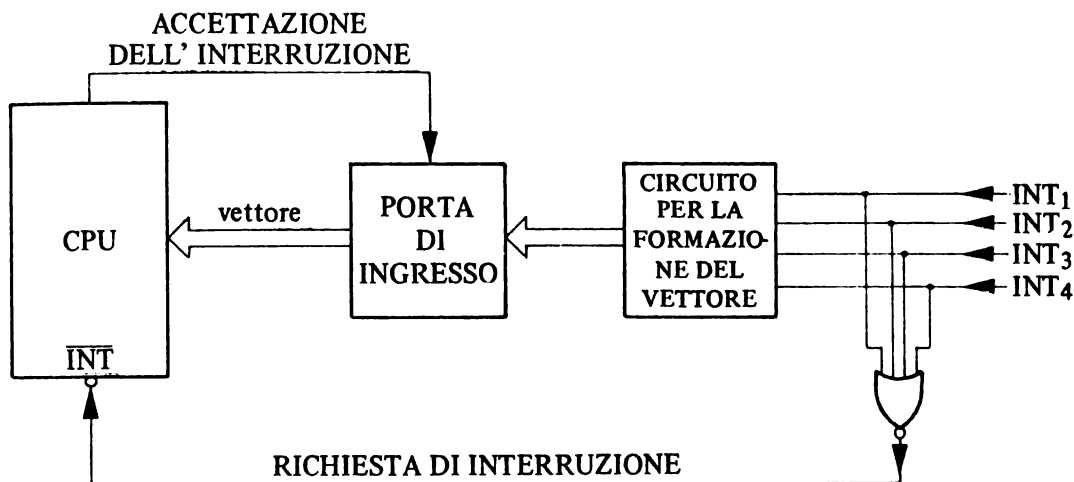



Fig. 4.6
Vettorizzazione delle interruzioni.

Alla richiesta di interruzione da parte di una periferica si ha, tramite la porta OR, l'attivazione dell'ingresso INT/ della CPU e contemporaneamente la formazione di un vettore di indirizzo, che è presentato ad una porta di ingresso.
Quando accetta l'interruzione il microprocessore acquisisce il contenuto della porta, dalla quale perciò deduce l'indirizzo di inizio della routine di servizio corrispondente.
Le modalità di formazione del vettore sono diverse a seconda del tipo di processore e delle particolari caratteristiche di gestione.
In genere si fornisce o la totalità o una parte dell'indirizzo di partenza della routine di servizio; in altri casi viene fornito il codice operativo di una particolare istruzione di CALL.
Il circuito per la formazione del vettore può presentare una certa complessità ed è in per questo motivo che si preferisce utilizzare dei circuiti integrati, costruiti allo scopo, i quali presentano anche il notevole vantaggio di essere programmabili, e quindi hanno la possibilità di poter variare il vettore fornito mediante semplici comandi da parte della CPU.

4.8. Priorità

Finora non si è mai affrontato il problema di come il microprocessore si deve comportare quando più periferiche effettuano contemporaneamente una richiesta di interruzione (per contemporaneamente si intende che più sorgenti effettuano la richiesta in un intervallo di tempo pari all'esecuzione di una istruzione, per cui il microprocèssore, al termine di essa e quando analizza la richiesta di servizio, trova più sorgenti di interruzione in attesa). Strettamente collegato a tale problema esiste poi quello che si verifica quando il microprocessore sta servendo una richiesta e ne viene attivata un'altra.
Anche in questo caso occorre stabilire come si possa ottenere una corretta ed efficiente gestione delle richieste: per risolvere queste situazioni, occorre stabilire delle priorità nei confronti delle diverse richieste.
Il criterio adottato è così organizzato: ad ogni sorgente di interruzione si assegna un livello di priorità; nel caso più richieste siano fatte contemporaneamente va servita per prima quella a priorità più elevata; quando il microprocessore sta servendo una interruzione la relativa routine può essere interrotta solo da quelle a priorità superiore.
Ovviamente si possono avere diverse soluzioni pratiche che dipendono dal tipo di microprocessore impiegato e dalle situazioni nelle quali si deve operare; allo scopo quasi tutti gli attuali microprocessori hanno la possibilità, via software, di abilitare o disabilitare la loro stessa capacità di rispondere alle richieste di interruzione.
Se nel microprocessore sono presenti più ingressi per le richieste, mediante una opportuna gestione delle abilitazioni e disabilitazioni è anche possibile una gestione elastica delle priorità nel senso che questa può essere variata a seconda delle situazioni che si presentano nel funzionamento di un sistema. Molto più di frequente ci si trova di fronte alla situazione in cui al microprocessore può essere presentata una sola richiesta in quanto esso possiede un unico ingresso per questo scopo: è allora necessario ricorrere ad un opportuno hardware esterno che permetta il desiderato cambiamento delle condizioni di priorità. Uno schema a blocchi utilizzabile potrebbe essere quello di fig. 4.7, in qui è indicata solamente la parte di un circuito con cui è possibile abilitare o meno una richiesta.
Come si può notare una richiesta da parte di un dispositivo provoca l'attivazione dell'ingresso INT/ solamente se è a valore logico alto il bit di abilitazione ad essa associato della porta di uscita.
Quest'ultima è gestita direttamente dall'unità centrale la quale pone i vari bit di abilitazione alti o bassi a seconda che voglia far transitare, o meno, la richiesta. Se, ad esempio, la CPU sta servendo una routine al massimo livello di priorità è inutile che essa sia interrotta al solo scopo di analizzare una richiesta a più basso livello, dato che sicuramente la CPU non mande-

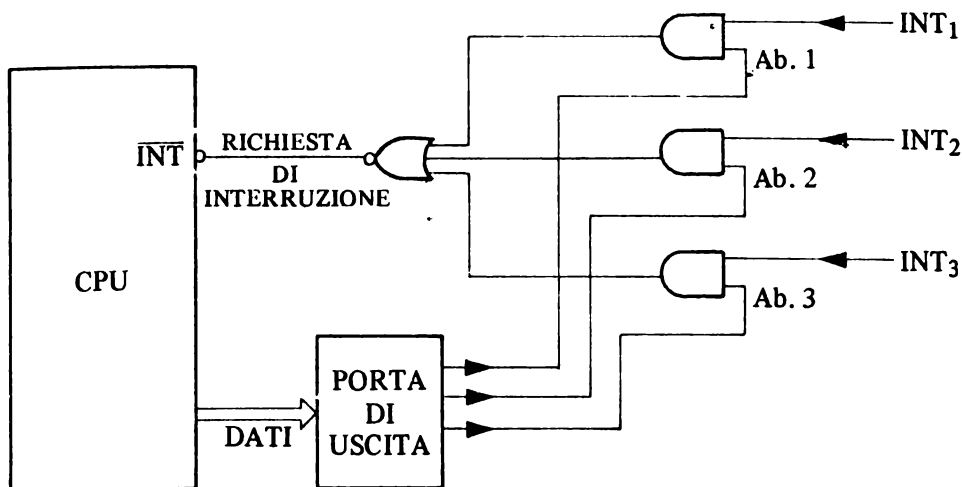



Fig. 4.7
Mascheratura delle richieste di interruzione.

rà in esecuzione la relativa routine: tra l'altro l'individuazione della sorgente provoca ovviamente una perdita di tempo.
Una volta eseguita la routine di servizio della richiesta ad elevata priorità la CPU procede, con una istruzione di scrittura sulla porta di uscita, alla riabilitazione delle altre richieste, per cui sarà di nuovo attivato l'ingresso INT/ se ne risulta pendente qualcuna.
Con lo schema adottato si possono anche ovviare a certi inconvenienti che si possono presentare se ci sono richieste troppo frequenti: infatti si può verificare che la routine di servizio di un dispositivo a priorità bassa sia continuamente interrotta da dispositivi a priorità superiore, per cui il primo rischia di essere servito in tempi eccessivamente lunghi. A tale inconveniente si può ovviare con una opportuna gestione delle abilitazioni; ad esempio si può fare in modo che una sorgente appena servita non sia riabilitata fin tanto che non siano state servite le rimanenti richieste pendenti: in questo modo si limita il tempo di attesa delle richieste a priorità bassa.
Ovviamente tale gestione richiede una adatta organizzazione del software, il quale, se presenta il vantaggio di una notevole flessibilità, ha anche l'inconveniente di richiedere un certo tempo per scrivere il registro di uscita prima che sia mandata in esecuzione la routine di servizio della richiesta in corso.

Come si è visto, nel caso di una organizzazione delle interruzioni a polling i livelli di priorità sono imposti via software dai criteri adottati per la scansione, per cui risulta possibile e sufficientemente semplice, una gestione elastica: ad esempio si possono variare i livelli di priorità. Si può anche dire che in una organizzazione a polling la priorità è stabilita dall'ordine con cui la scansione analizza i bit di stato.
Una analoga organizzazione, però realizzata via hardware, è la tecnica che prende il nome di "daisy chain", rappresentata schematicamente in fig. 4.8; in essa ogni periferica, oltre all'uscita per la richiesta di interruzione, possiede un ingresso IEI (Interrupt enable input) ed un'uscita IEO (Interrupt enable output).

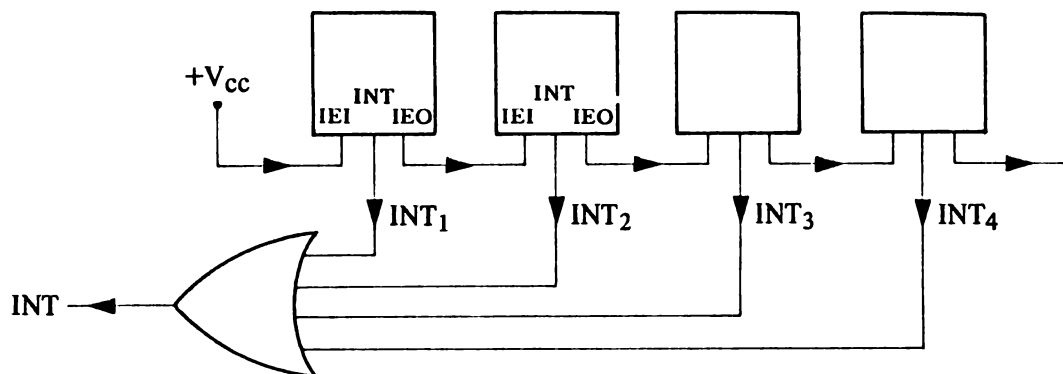



Fig. 4.8
Daisy-chain.

L'ingresso IEI del primo dispositivo della catena si trova a valore logico alto e tale dispositivo possiede la più elevata priorità, mentre i rimanenti hanno una priorità decrescente che dipende dalla posizione da essi occupata nella catena.
Le relazioni che legano l'ingresso IEI alle uscite IEO ed INT sono le seguenti: se IEI è a valor logico basso, anche IEO assume tale valore ed il dispositivo non può attivare la sua uscita INT per la richiesta di interruzione; se IEI è a valor logico alto il dispositivo può attivare la sua uscita INT, e contemporaneamente deve porre a livello logico basso la sua uscita IEO.
Ne risulta quindi che un dispositivo che presenti attiva la sua uscita INT

impedisce a tutti i successivi dispositivi della catena di richiedere le interruzioni: in questo modo è automaticamente realizzato via hardware un criterio di priorità.
La tecnica del daisy chain è molto utilizzata, ma presenta l'inconveniente, come sempre nelle strutture hardware, di non permettere la variazione delle priorità.
Lo schema relativo alla generazione di segnali di una catena a daisy-chain potrebbe essere quello riportato nella fig. 4.9.

Fig. 4.9
Rete per la generazione dei segnali nella catena in daisy-chain.

Come si è visto è molto comodo aver la possibilità di abilitare o disabilitare la richiesta di interruzione presente in ogni sorgente. Anche le CPU hanno in genere questa possibilità, almeno per alcuni ingressi di interruzioni, e di solito si ottiene anche una disabilitazione automatica una volta accettata una richiesta. Se ciò non si verificasse, dopo aver accettata una richiesta e mandato in esecuzione il relativo programma di gestione, la

CPU non avrebbe la certezza di procedere al salvataggio del suo stato in quanto il segmento di programma a ciò dedicato potrebbe a sua volta essere interrotto da una richiesta a livello superiore con conseguente perdita di informazioni.

4.9. Velocità di intervento

Dall'istante in cui si attiva una richiesta di interruzione a quando va in esecuzione la relativa routine di servizio passa un certo intervallo di tempo, detto tempo di intervento.
Ad esso contribuiscono tre distinti intervalli temporali, caratterizzati dalle operazioni che il microprocessore vi compie: il primo intervallo, detto anche tempo di latenza, è quello che intercorre dall'istante in cui si attiva il segnale di interruzione a quello in cui il microprocessore analizza tale segnale; tale intervallo è, nella peggiore delle ipotesi, pari alla durata della più lunga istruzione eseguibile. Il secondo è costituito dal tempo necessario per mandare in esecuzione la routine di servizio: è quello impiegato per salvare il program counter, e calcolare il nuovo indirizzo; esso evidentemente varia a seconda del microprocessore e delle caratteristiche dell'interruzione. Il terzo intervallo di tempo è quello che il microprocessore impiega per eseguire quelle istruzioni che hanno il solo compito di salvare lo stato dell'unità centrale per permettere una corretta ripresa del programma interrotto. Anche in questo caso si possono avere diverse situazioni che dipendono essenzialmente dalle caratteristiche del programma di gestione di quella particolare richiesta di interruzione.

4.10. Inconvenienti delle interruzioni

La gestione di periferiche mediante richiesta di interruzione permette di ottenere indubbi vantaggi e migliori prestazioni rispetto alla tecnica a polling, ma presenta anche alcuni svantaggi che devono essere tenuti presenti per una scelta della struttura da dare al microelaboratore.
Come si è visto una gestione con interruzione richiede una struttura hardware più complessa, rispetto a quella necessaria nella gestione a polling: è questo un aspetto che ha una importanza sempre minore in quanto generalmente si utilizzano dei circuiti integrati realizzati allo scopo ed a volte inseriti negli stessi componenti periferici della famiglia del microprocessore.
L'aspetto più grave riguarda invece il software ed è dovuto fondamentalmente alla natura asincrona e casuale della interruzione, proprietà che ne costituisce d'altra parte una caratteristica essenziale. Ciò comporta una notevole difficoltà di verifica di programmi funzionanti a interruzione in quanto non è agevole controllare tutte le possibili situazioni che potrebbero

verificarsi. Si devono allora adottare dei particolari criteri nella stesura delle subroutine di servizio in modo da evitare situazioni di errore, che peraltro, per la loro natura casuale, sono difficilmente diagnosticabili.
La tendenza attuale è quella di usare le interruzioni solamente quando strettamente necessario, come ad esempio per servire segnalazioni di allarme, per le quali è richiesto un tempo di intervento il più piccolo possibile.

4.11. Gestione delle interruzioni nel microprocessore Z80

Il microprocessore Z80 ha due ingressi ai quali le periferiche possono presentare richiesta di interruzione: NMI/ ed INT/. Il primo, Non Mascherabile Interrupt, ha la priorità più elevata ed è sempre abilitato; è in genere utilizzato per le richieste di interruzione nei casi in cui deve essere assolutamente garantito l'intervento del processore, interrompendo qualsiasi tipo di programma in esecuzione. E' ad esempio usato per segnalare in anticipo una caduta della tensione di alimentazione: se l'intervento è tempestivo, il microprocessore riesce a salvare lo stato della macchina per poter riprendere correttamente l'elaborazione quando si ripristino i valori corretti della tensione stessa.
L'altro ingresso disponibile, INT/, serve invece per la gestione normale: esso può essere mascherabile nel senso che può essere disabilitato sia al verificarsi di determinate situazioni, sia da programma con una istruzione "DI" (Disable Interrupt). Inoltre per esso c'è la possibilità di stabilire da programma uno tra tre modi diversi di gestione delle interruzioni.
Affinché il processore Z80 accetti una richiesta di interruzione esso si deve trovare nella funzione di master, cioè non deve aver ceduto ad altri dispositivi il controllo dei bus: questa è l'unica condizione che deve essere soddisfatta nel caso di NMI/.
Per la richiesta mascherabile, INT/, oltre a quanto appena detto, il microprocessore deve essere stato precedentemente abilitato ed accettare tale richiesta, tramite l'esecuzione della istruzione "EI" (Enable Interrupt).
Esistono tre situazioni, che possono interrompere l'esecuzione di un programma, alle quali il microprocessore reagisce in modo però da essere successivamente in grado di riprendere correttamente l'esecuzione arrestata. In ordine di priorità decrescente esse sono:

- richiesta del rilascio del controllo del bus (BUSRQ/);
- richiesta di interruzione non mascherabili (NMI/);
- richiesta di interruzione mascherabile (INT/).

Mentre la richiesta BUSRQ/ è controllata ogni ultimo periodo di clock di un ciclo macchina, le altre due lo sono solamente all'ultimo periodo del-

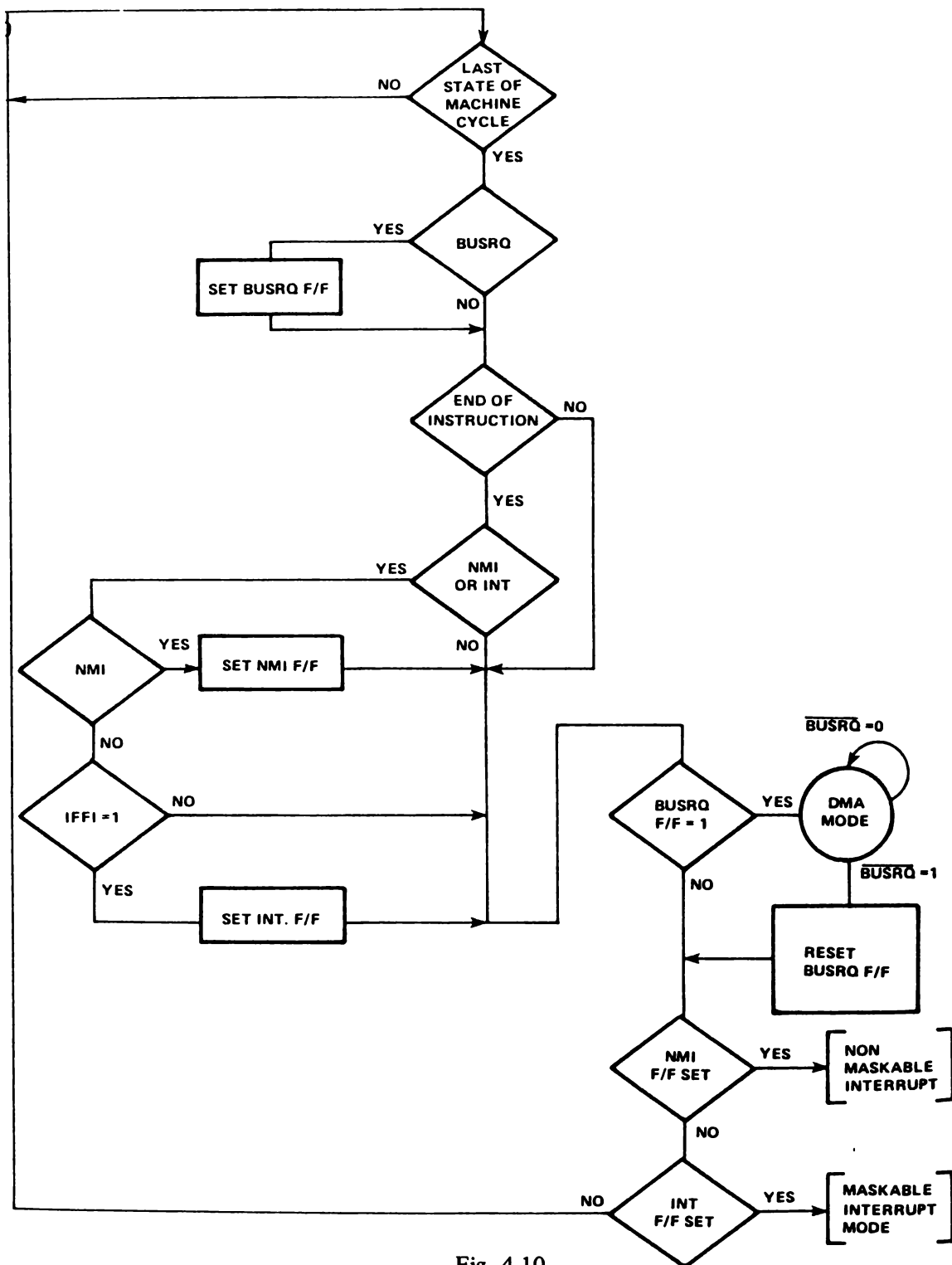



Fig. 4.10
Diagramma di flusso della gestione delle interruzioni nel microprocessore Z80.
(Mostek, Z80 Microcomputer Data Book, 1981).

l'ultimo ciclo macchina di ogni istruzione. Un diagramma di flusso delle varie attività che il processore mette in atto per la gestione di queste richieste è riportato nella fig. 4.10.
Come si può notare in essa si fa riferimento a dei circuiti bistabili (Flip-Flop), interni al microprocessore, mediante i quali è memorizzata la segnalazione di richiesta esterna.
La possibilità di abilitazione o meno delle richieste di interruzione da parte del programma comporta qualche differenza che ha anche conseguenze nella struttura hardware esterna. Infatti lo Z80 analizza le richieste solamente se è abilitato a questa attività; poiché la disabilitazione è automatica dopo ogni accettazione, nel caso di più richieste ci si può trovare nella situazione che una richiesta trovi già il microprocessore impegnato con una precedente, per cui essa non è memorizzata da parte dell'unità centrale. E' per questo motivo che per le interruzioni mascherabili è necessario provvedere con dell'hardware esterno alla conservazione della richiesta finché non viene servita.
Sempre relativamente alle interruzioni non mascherabili esistono due FF interni allo Z80, IFF1 e IFF2, che possono essere settati o resettati da programma ed il cui uso verrà descritto in seguito.

4.12. Interruzione non mascherabile

I segnali che interessano in un ciclo relativo ad una richiesta di interruzione non mascherabile ed alla sua accettazione sono riportati nella fig. 4.11, (si fa l'ipotesi che siano soddisfatte le diverse condizioni affinché la CPU accetti tale richiesta).
Il microprocessore, sul fronte di salita dell'ultimo periodo di clock dell'ultimo ciclo macchina dell'istruzione in esecuzione, testa lo stato dell'ingresso NMI/. Si supponga che esso sia stato attivato, cioè si trovi a livello logico basso: in queste condizioni il successivo ciclo macchina inizia come un normale ciclo di fetch: è infatti inviato sul bus degli indirizzi il contenuto del program counter e sono attivati i segnali M1/, MREQ/, e RD/.
Se si osserva quanto accade successivamente, non si nota alcuna differenza apparente rispetto ad un normale ciclo di fetch: infatti il microprocessore non segnala all'esterno che sta intraprendendo le azioni necessarie per la gestione di una richiesta di interruzione. D'altra parte tale informazione non è necessaria in quanto una volta fatta una richiesta di interruzione non mascherabile sicuramente il microprocessore provvede alla sua accettazione. Quindi anche se la memoria è attivata per fornire il codice operativo individuato dal contenuto del bus degli indirizzi, il processore non utilizza tali informazioni, ma si predispone internamente per mandare in esecuzione la routine di servizio dell'interruzione.

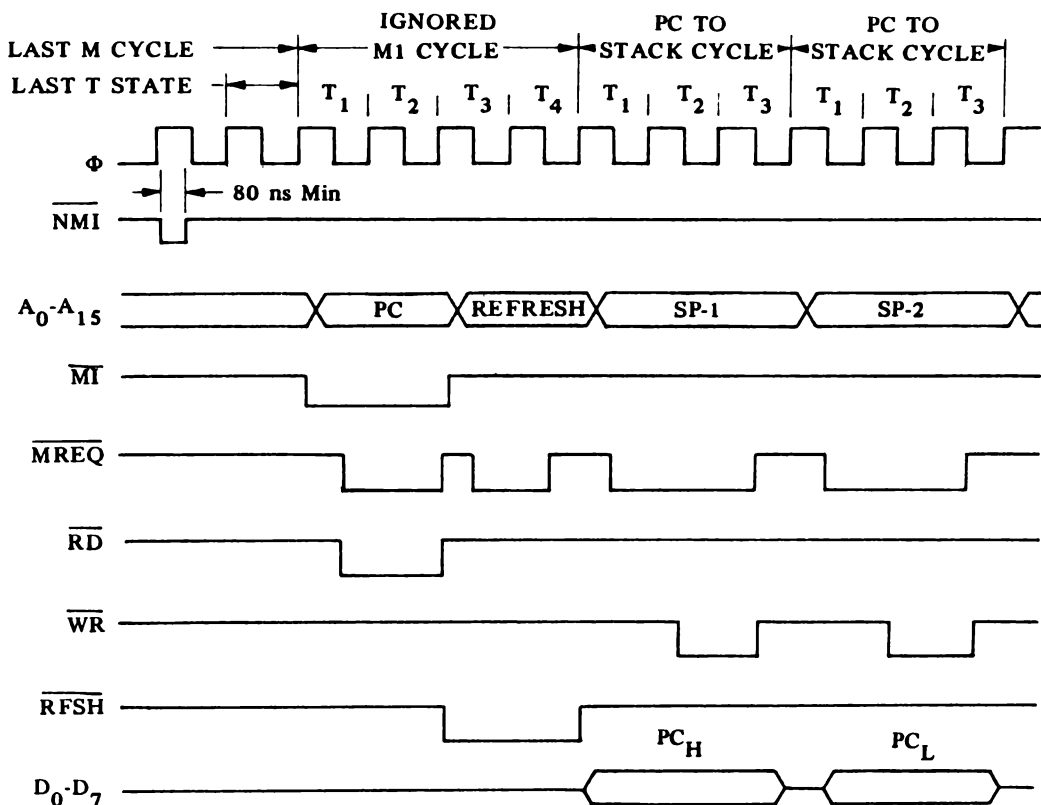



Fig. 4.11
Diagramma temporale nella gestione delle interruzioni non mascherabili (NMI).

La prima azione che lo Z80 esegue in modo diverso rispetto ad un ciclo di fetch consiste nel non incrementare il program counter, per cui questi contiene l'indirizzo della istruzione successiva a quella appena ultimata; è eseguita invece la fase di rinfresco delle eventuali memorie dinamiche presenti. In due cicli successivi la CPU provvede automaticamente a salvare il contenuto del program counter nella memoria RAM all'indirizzo indicato dallo stack pointer.
Essa procede quindi a caricare nel program counter un valore fisso, 0066H, per passare solo ora alla normale fase di fetch relativa a tale indirizzo. E' perciò necessario che la routine di gestione all'interruzione non mascherabile inizi in quella posizione di memoria.
Un'altra azione intrapresa nella fase di accettazione di un'interruzione non mascherabile è la gestione dei due Flip Flop IFF1 e IFF2. Tali FF hanno

normalmente lo stesso valore: infatti una istruzione di Set o Reset li posiziona entrambi allo stesso modo. Nel caso invece dell'accettazione dell'interruzione non mascherabile il microprocessore azzera il contenuto di IFF1 dopo averlo trasferito in IFF2.
In tal modo l'unità centrale è disabilitata dall'accettare nuove richieste di interruzione, ma contemporaneamente è conservato il valore precedente di IFF1 e ciò permette di ripristinare lo stato dell'unità centrale alla fine della routine di servizio.
Quest'ultima deve terminare con una istruzione particolare di ritorno al programma interrotto, "RETN" (return da interrupt non mascherabile): con essa non solo si ripristina il valore del program counter, conservato nello stack, ma anche il valore di IFF1. La sequenza delle attività svolte nella gestione di una interruzione non mascherabile è riportata nella fig. 4.12.

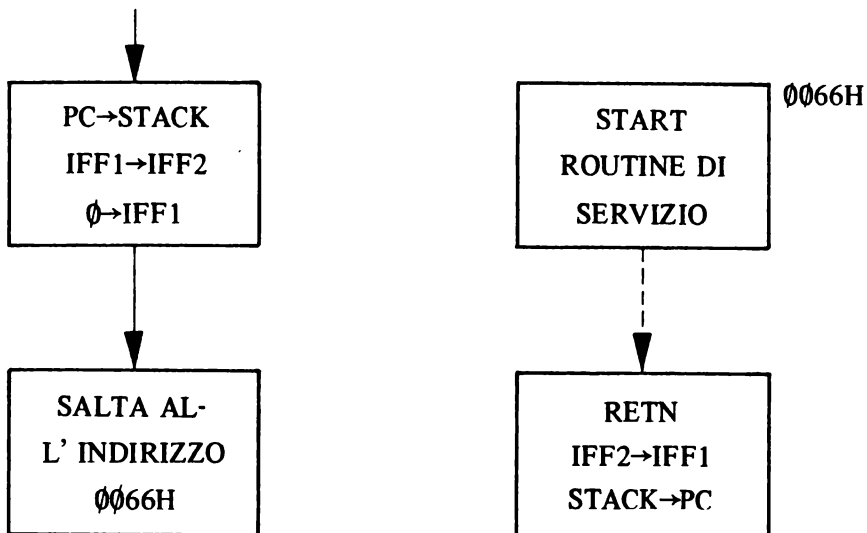



Fig. 4.12
Diagramma di flusso relativo alla richiesta di interruzione NMI.

In conclusione per la gestione di una richiesta di interruzione non mascherabile non è necessario dell'hardware esterno ad eccezione di quello che serve per formulare la richiesta stessa.

4.13. Interruzioni mascherabili

Per quanto riguarda invece le interruzioni mascherabili il microprocessore Z80 può essere programmato in modo da gestirle con tre modalità differenti: esistono infatti tre istruzioni distinte, "IM0" "IM1" ed "IM2", le quali permettono di imporre al microprocessore di gestire le richieste di interruzione rispettivamente nel modo 0, 1 o 2.
Il modo di funzionamento 1 è molto simile a quello visto nel caso delle interruzioni non mascherabili: supposte infatti soddisfatte le condizioni per accettare la richiesta di interruzione, il microprocessore provvede al salvataggio del program counter, forzandone successivamente il suo contenuto al valore 38H. In questa posizione deve necessariamente iniziare la routine di gestione per tale tipo di interruzione.
La sequenza delle attività espletate nella gestione di questo modo di interruzione è indicata nella fig. 4.13.

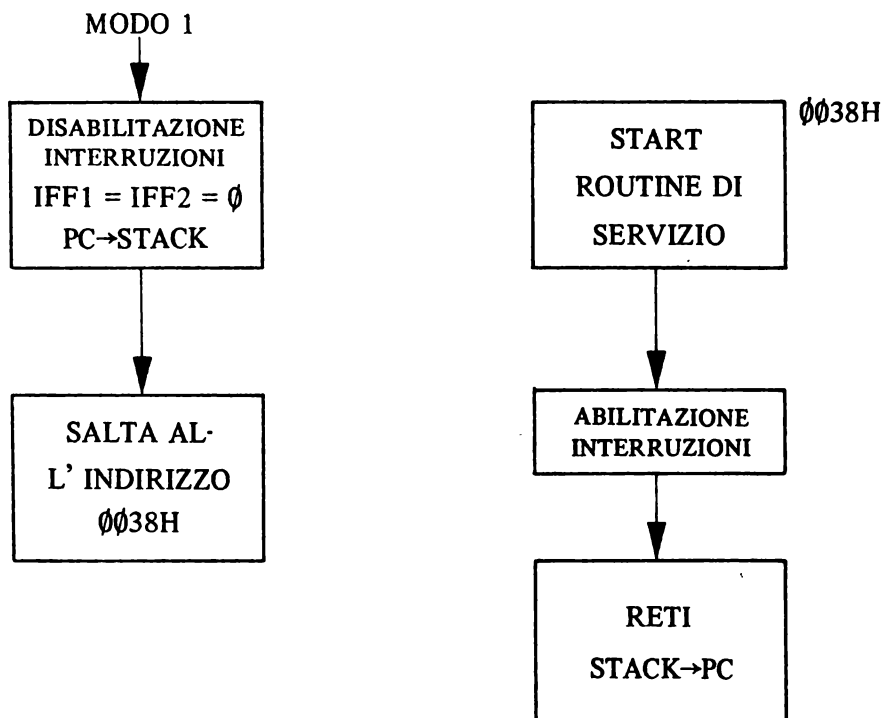



Fig. 4.13
Diagramma di flusso relativo alla richiesta di interruzione in modo 1.

Come si può notare in questo caso le interruzioni sono disabilitate ponendo entrambi i FF, IFF1 e IFF2, a zero. La fine della routine è individuata dalla istruzione di "RET" mediante la quale si ha il ripristino del program counter inizialmente salvato nello stack.

L'abilitazione delle interruzioni si attua da programma, con l'istruzione EI, che può essere posta all'interno del programma di servizio, o in altre parti, a seconda delle particolari esigenze di elaborazione; in genere si pone come ultima istruzione prima di RET.
Lo Z80 non analizza lo stato delle linee di interruzione all'ultimo ciclo macchina di tale istruzione. Il motivo di tale comportamento è che così facendo si evita di aumentare in modo incontrollabile, e quindi con pericolo di overflow, la dimensione dello stack. Infatti se lo Z80 non si comportasse in questo modo, alla fine dell'esecuzione di EI si avrebbe il microprocessore immediatamente abilitato, col che, nel caso di una richiesta pendente, esso procederebbe subito alla sua accettazione, senza eseguire perciò l'istruzione di RET e quindi non scaricando lo stack. Nella nuova routine di servizio potrebbe ripetersi la stessa situazione per cui lo stack continuerebbe ad aumentare, per giunta senza alcuna utilità.
Il modo di funzionamento 0 è identico al modo di gestione delle interruzioni nel microprocessore 8080: infatti la principale ragione per cui si ha nello Z80 tale modo di funzionamento è che si è voluto conservare la compatabilità al software del processore 8080.
Nel modo 0 il dispositivo esterno che ha richiesto l'interruzione può fornire, in sincronismo con il funzionamento del microprocessore, una istruzione di Restart n, "RSTn" che è una istruzione di chiamata a subroutine ad indirizzi prefissati e cioè ad uno degli otto indirizzi individuati dal valore di tre bit del codice operativo.
Se si vuole invece saltare in qualunque posizione di memoria è necessario fornire un indirizzo costituito da 16 bit, assieme al codice operativo della istruzione CALL, per cui in totale sono necessari ben 24 bit. Utilizzando le istruzioni di RSTn sono possibili solamente 8 sorgenti di richieste distinte e le relative routine devono iniziare in posizioni di memoria prefissate; si noti inoltre che è necessario dell'hardware esterno per provvedere a fornire il codice operativo dell'istruzione RST.
Il modo di funzionamento 2 è quello che permette una maggior flessibilità; anche in questo caso è l'hardware esterno che deve fornire delle informazioni, che consistono in un vettore di 8 bit, mediante il quale lo Z80 può individuare l'inizio della routine di servizio, che si può trovare in qualsiasi posizione di memoria.
Gli 8 bit che sono forniti dall'hardware esterno costituiscono la parte meno significativa di un indirizzo; la parte più significatica si trova su un apposito registro interno allo Z80, che deve essere stato precedentemente caricato da programma. Con questi due byte la CPU forma un indirizzo mediante il quale essa individua una posizione di memoria la quale, assieme alla posizione successiva, contiene l'effettivo indirizzo di inizio della routine di servizio.

Il reperimento dell'indirizzo di partenza cercato è rappresentato in forma schematica nella fig. 4.14.

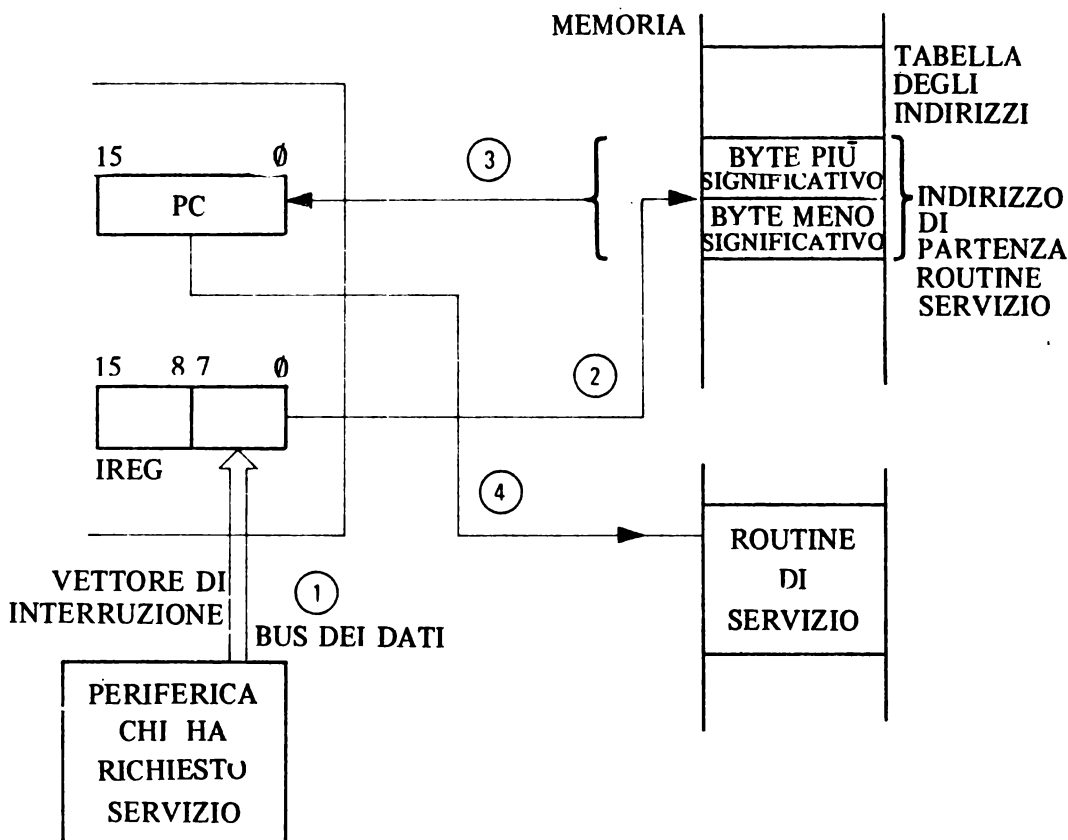



Fig. 4.14
Reperimento dell'indirizzo della routine di interruzione da parte dello Z80.

Come detto è necessaria una struttura hardware esterna per fornire la parte meno significativa dell'indirizzo della tabella, mentre tutte le altre operazioni, in particolare il salvataggio del program counter e suo successivo caricamento con l'indirizzo trovato in memoria è automaticamente gestito dal microprocessore. E' necessaria anche una addatta preparazione software affinché nelle posizioni previste si trovino le informazioni necessarie.

L'unica osservazione da fare è che il vettore fornito da componenti periferici della famiglia Z80 è composto solamente da 7 bit significativi in quanto il bit meno significativo deve essere necessariamente 0.
Il diagramma temporale della fase di richiesta e di parte della fase di accettazione è riportato nella fig. 4.15.

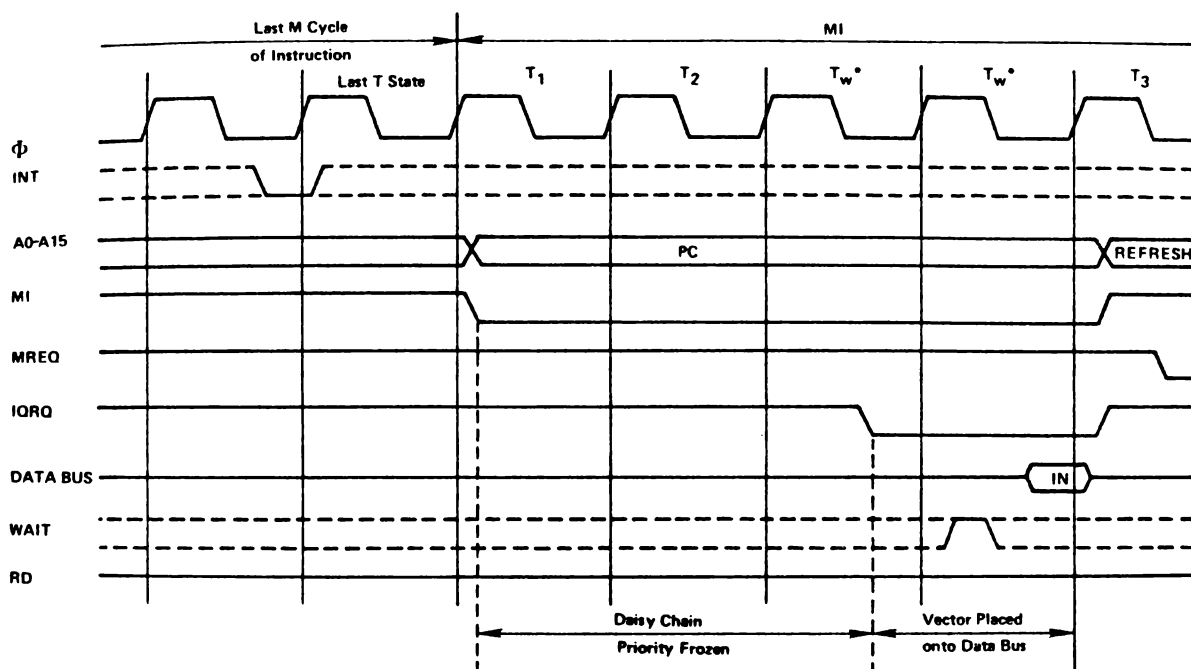



Fig. 4.15
Ciclo di richiesta e di accettazione di una interruzione mascherabile.
(Mostek, Z80 Microcomputer Data Book, 1981)

Anche in questo caso il microprocessore analizza lo stato dell'ingresso INT/ sul fronte di salita dell'ultimo periodo di clock dell'ultimo ciclo macchina di ogni istruzione. Se tale segnale è attivo, e nell'ipotesi che siano soddisfatte le varie condizioni necessarie per l'accettazione, il microprocessore inizia il ciclo successivo durante il quale informa l'esterno che sta accettando la richiesta. Tale ciclo non è, come nel modo 1, un normale ciclo di fetch non sfruttato; esso presenta invece qualche particolarità: la CPU invia inizialmente, come al solito, il contenuto del program counter, che è

puntato all'istruzione che non sarà eseguita, ed attiva il solo segnale M1/. Il successivo periodo T_2 non vede il microprocessore impegnato in alcuna attività con l'esterno; invece di passare direttamente al periodo T_3 sono inseriti automaticamente due periodi di attesa. Durante questi è attivato anche il segnale IORQ/: è la contemporanea attivazione del segnale M1/ ed IORQ/ che deve essere interpretata dall'esterno come il segnale di accettazione della interruzione da parte del microprocessore.

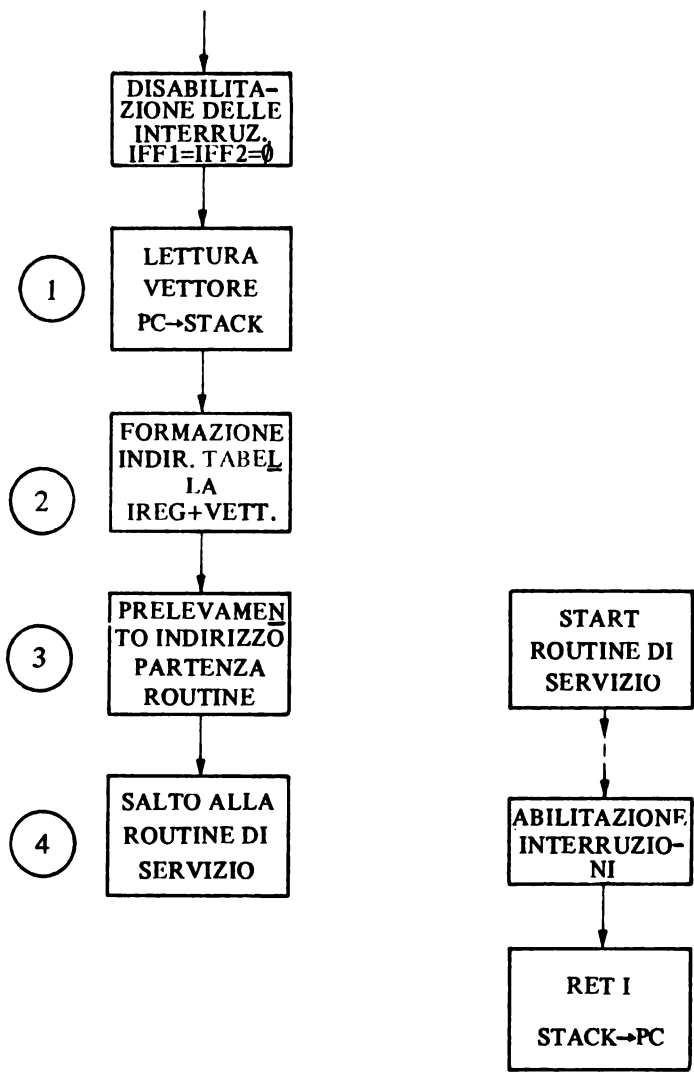



Fig. 4.16
Diagramma di flusso relativo all'accettazione delle richieste di interruzione in modo 2.

Nel successivo fronte di salita del clock, nel periodo T_3, il microprocessore procede alla acquisizione della informazione presente sul bus dei dati, la quale è interpretata come il vettore per il riconoscimento della sorgente della interruzione. Come sempre, se l'hardware esterno non è in grado di fornire in tempo utile le informazioni richieste, possono essere introdotti degli stati di attesa aggiuntivi in modo da rendere compatibili le velocità di funzionamento.
Nei due successivi cicli macchina lo Z80 procede a memorizzare il contenuto del program counter nello stack e successivamente in altri due cicli esso deduce l'indirizzo di partenza della routine di servizio. Tale indirizzo viene posto nel program counter e si procede quindi con la fase di fetch della prima istruzione della routine. In totale sono necessari 5 cicli macchina; uno di fetch particolare, altri due di scrittura di memoria seguiti da due di lettura.
Il diagramma di flusso relativo al modo 2 è riportato nella fig. 4.16.
Anche in questo caso si ha automaticamente la disabilitazione delle interruzioni all'accettazione di una richiesta; l'abilitazione può avvenire con le stesse modalità già descritte.
Il ritorno al programma interrotto avviene in questo caso con una istruzione particolare "RETI" (return from interrupt) la cui funzione è identica a quella di una istruzione RET, ma presenta un codice operativo diverso e serve per la gestione delle priorità, come sarà illustrato fra breve.

4.14. Gestione delle priorità nello Z80

Se si utilizzano i modi di funzionamento 0 e 1 per le interruzioni, nel caso si debbano risolvere delle priorità si deve ricorrere a dei circuiti esterni, come quelli indicati in precedenza, naturalmente adattandoli alle caratteristiche dello Z80.
Con il modo 2 invece si hanno delle notevoli facilitazioni in quanto ogni periferica della famiglia dello Z80 è predisposta a generare delle richieste di interruzione con il microprocessore funzionante in tale modo ed inoltre è prevista la gestione delle priorità adottando la tecnica del daisy-chain.
Ogni componente della famiglia del microprocessore Z80 presenta i tre segnali per la gestione delle interruzioni: INT/, IEI e IEO. Il primo serve per segnalare all'unità centrale la richiesta, mentre gli altri due sono utilizzati per risolvere le priorità, e si comportano con le regole a suo tempo viste, caratterizzate ovviamente da precise relazioni temporali che tengono conto del particolare modo di funzionamento dello Z80.
Uno schema logico del circuito necessario per la gestione delle interruzioni, presente in qualsiasi periferica della famiglia Z80 in grado di generare interruzioni, è quello riportato nella fig. 4.17.

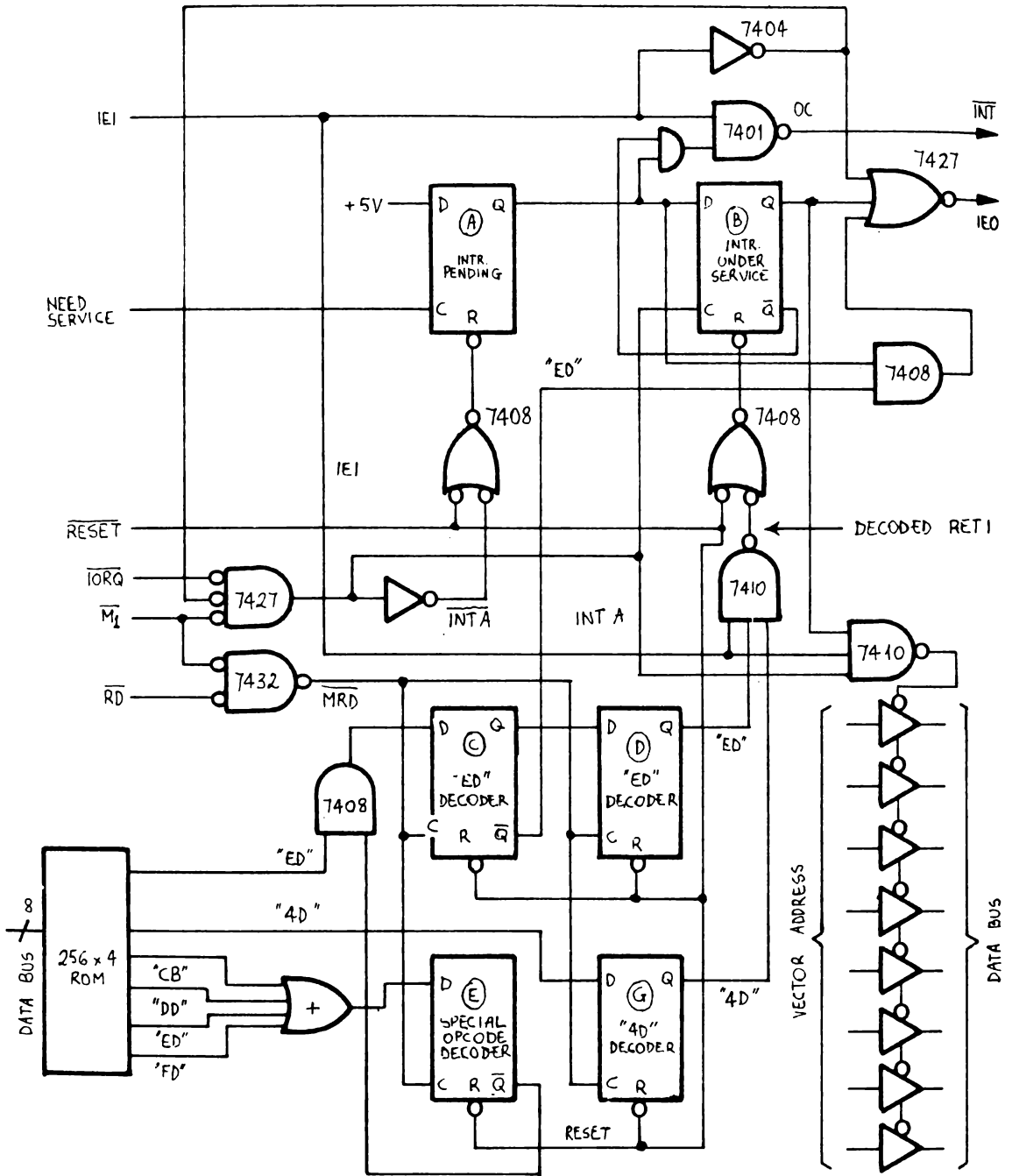



Fig. 4.17
Schema del circuito per la gestione della richiesta di interruzione nel modo 2.
(Zilog, Z80 - Technical Manual)

In esso si possono distinguere quattro parti funzionali:

1) quella relativa alla logica di generazione del segnale di richiesta di interruzione INT/;

2) quella che genera il segnale IEO per la risoluzione delle priorità nella catena daisy-chain;

3) quella per l'invio del vettore di interruzione sul bus dei dati;

4) quella di rivelazione del termine della routine di servizio.

I segnali di ingresso necessari al funzionamento di tale circuito possono essere suddivisi in tre gruppi:

a) i segnali di controllo, per la sincronizzazione e l'attivazione del circuito, inviati dalla CPU (IORQ/, M1/, RD/) o necessari alla inizializzazione (RESET);

b) il segnale che serve per la risoluzione delle priorità IEI;

c) i segnali presenti sul bus dei dati.

La causa di richiesta di servizio può essere diversa a seconda del tipo di dispositivo e del modo in cui è stato programmato: può essere la presenza di un dato pronto all'ingresso (PIO) o lo scadere di un certo intervallo di tempo (CTC) o la fine di un trasferimento (DMAC), ecc.
Tale segnalazione è indicata nello schema di fig. 4.17 con l'indicazione di "necessità di servizio" e si deve considerare attiva quando si porta dal valor logico basso a quello alto.
Si supponga che inizialmente sia stato attivato il segnale di ingresso RESET, per cui tutti i flip flop presenti nel circuito si trovano in una ben definita condizione, cioè con le uscite Q a livello logico basso.
Quando il segnale di "necessità di servizio" diventa attivo, esso provoca la commutazione del flip-flop A, di interruzione in attesa, cioè la sua uscita Q si porta al livello logico alto.
Dato che l'uscita del flip flop B, "interruzione in servizio", ha valore logico basso, Qb = 0, se il segnale IEI si trova a livello logico alto, si ha l'attivazione dell'uscita INT/.
Per quanto riguarda il segnale di uscita IEO, il suo valore si può ricavare dalla relazione:

$$IEO = \overline{Qa \cdot Qc/ + Qb + IEI/}.$$

Dato che per le condizioni iniziali risulta Qc/ = 1, Qb = 0, IEI/ = 0, la necessità di servizio provoca Qa = 1 e quindi il segnale IEO si porta a livello logico basso.
Poiché i dispositivi sono collegati in cascata tale valore di IEO si propaga nella catena e ciò provoca che un'eventuale necessità di servizio da parte

di un dispositivo che segue nella catena non può generare una attivazione del segnale di uscita INT/, dato che per tale dispositivo il segnale IEI è a valor logico basso, ma rimane solo conservata al suo interno nel flip flop A "interruzione in attesa".
In conclusione nessuno dei dispositivi successivi è in grado di generare una interruzione e tutti i loro segnali IEI ed IEO si trovano a livello logico basso.
Una volta attivata, l'uscita INT/ nel dispositivo in esame provoca la richiesta di servizio all'unità centrale: lo Z80 risponde alla richiesta con l'accettazione della interruzione rendendo attivi contemporaneamente i due segnali di controllo M1/ e IORQ/. L'attivazione di questi due segnali di ingresso fa si che il FF B si porti con la sua uscita a livello alto, Qb = 1, e che il FF A commuti, per cui Qa = 0.
Questa nuova situazione provoca l'abilitazione del buffer di uscita, che precedentemente era nello stato di alta impedenza, per cui sul bus dei dati è inviato il vettore di interruzione.
Si noti che per l'attivazione di tale buffer devono essere soddisfatte le seguenti condizioni: IORQ/ = 0, M1 = 0, IEI = 1, e Qb = 1: le prime due assicurano che l'unità centrale sta accettando una richiesta di interruzione, l'ultima assicura che il dispositivo ha una richiesta pendente, mentre la terza permette solo al dispositivo, che nella catena si trova con IEI alto, di inviare il vettore.
L'attivazione di Qb ha un'ulteriore conseguenza e precisamente essa fa si che il segnale di uscita, INT/, si porti a livello logico alto: da ciò si deduce che finché un dispositivo è sotto servizio non è in grado di generare una ulteriore richiesta di interruzione.
Come si è detto in precedenza l'ultima istruzione di una routine di servizio di una interruzione è RETI (return from interrupt) la cui azione, come in una normale istruzione RET, è quella di ritornare al programma interrotto; l'unica differenza fra le due istruzioni è nel codice operativo: RETI presenta il codice EDH seguito da 4DH, RET il codice C9H.
La rete per la gestione del daisy-chain, che è collegata al bus dei dati, oltre ad inviare in esso il vettore di interruzione, può controllare quando la routine di servizio ha termine: la memoria ROM in essa contenuta rivela se sul bus dei dati è presente il codice EDH. Se ciò si verifica durante un ciclo di fetch (M1/ attivo assieme a RD/) il FF C viene settato per cui Qc/ si porta a livello logico basso.
Poiché il FF B ha ancora la sua uscita alta, il riconoscimento del codice EDH non provoca nessun altro effetto visibile. Nel successivo ciclo di fetch il circuito riconosce il codice 4DH dell'istruzione RETI e poiché IEI = 1, si ha il reset del flip-flop B, per cui l'uscita IEO si porta a livello logico alto e permette quindi anche a dispositivi che si trovano nelle posizioni successive nella catena di richiedere delle interruzioni.

Nell'analisi appena fatta si è sempre considerato IEI a livello logico alto e cioè il dispositivo era quello a più elevata priorità, oppure nessuno dei dispositivi presenti nella catena, a priorità più elevata, aveva fatto richiesta di interruzione.
Si supponga ora che la richiesta di servizio del dispositivo B sia stata accettata, ne sia in esecuzione la relativa routine di servizio e l'unità centrale sia stata riabilitata ad accettare le interruzioni. Se un dispositivo a priorità più elevata, ad esempio A di fig. 4.18, fa richiesta di interruzione, cioè attiva la sua uscita INT_A/, è interrotto il programma di servizio di B e mandata in esecuzione la routine relativa ad A.

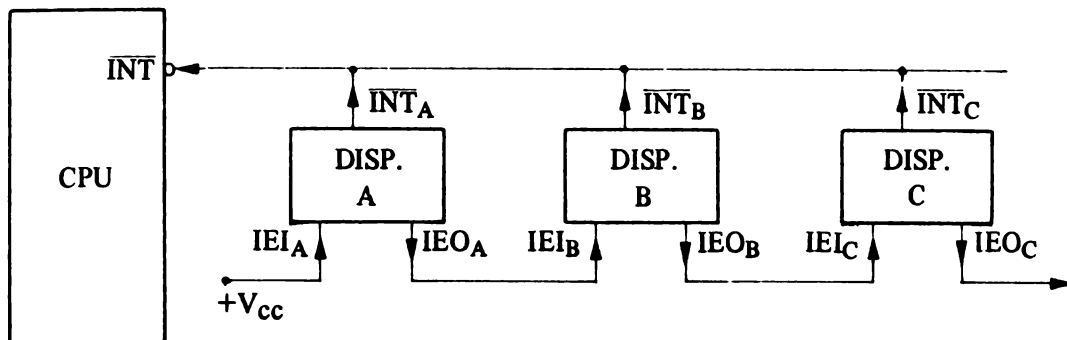



Fig. 4.18
Catena di interruzione nello Z80.

L'attivazione di INT_A/ ha provocato il passaggio al valor logico basso del segnale IEO_A e quindi di IEI_B: il dispositivo B si trova ora in una condizione diversa da quella prima vista. Al termine della routine di servizio di A, quando si è nel ciclo di fetch della istruzione RETI, i segnali di controllo IORQ/ ed M1/ fanno si che il dispositivo A porti al valor logico alto la sua uscita IEO_A, mentre B, che all'arrivo del codice di RETI aveva il suo IEI basso, non modifichi il valore della sua uscita IEO_B. Solo al termine della routine di servizio di B saranno abilitati a richiedere interruzione i dispositivi a priorità più bassa di B, nella catena.
Un'altra situazione che può verificarsi è che il dispositivo B di fig. 4.18 sia in servizio ma in questo caso l'unità centrale non sia stata riabilitata ad ac-

cettare nuove interruzioni. Se ora il dispositivo A necessita di servizio esso attiva INT/ ed il suo IEO va a livello logico basso per cui tutti i successivi IEI della catena, compreso quindi quello del dispositivo B, si portano a tale livello, ma non viene accettata la richiesta di interruzione di A dato che l'unità centrale non è abilitata.
Quando ha termine la routine di servizio del dispositivo B, questi si trova in una situazione analoga alla precedente ma con la differenza che in questo caso l'istruzione RETI esaminata è relativa alla sua stessa routine.
Si noti che anche il dispositivo A analizza il bus dei dati e verifica la presenza del codice EDH; esso si trova nelle condizioni di avere IEI al valore logico alto, Qa = 1 e Qb = 0. Si verifica allora che quando è presente il codice EDH il segnale IEO di A si porta al valore logico alto per cui si porta a tale livello anche il segnale IEI del dispositivo B. Al successivo ciclo di fetch il circuito B rivela il codice operativo 4DH per cui il relativo IEO viene portato anch'esso al valor logico alto, in quanto IEI si trova a tale livello, mentre il dispositivo A non può agire in quanto il suo FFA si trova con l'uscita Qa alta. Non appena termina la fase di fetch di 4DH il FFC porta la sua uscita Qc = 1 per cui il dispositivo A può porre a livello basso il suo IEO ed attendere l'accettazione della sua richiesta da parte dell'unità centrale.
In fig. 4.19 sono indicati gli istanti di attivazione o meno dei segnali IEI ed IEO nel riconoscimento di una istruzione di RETI.
Un'altra situazione che si può verificare è quella in cui essendo il dispositivo C sotto servizio, con interruzioni non abilitate, sia il dispositivo A che B attivino la richiesta di interruzione.

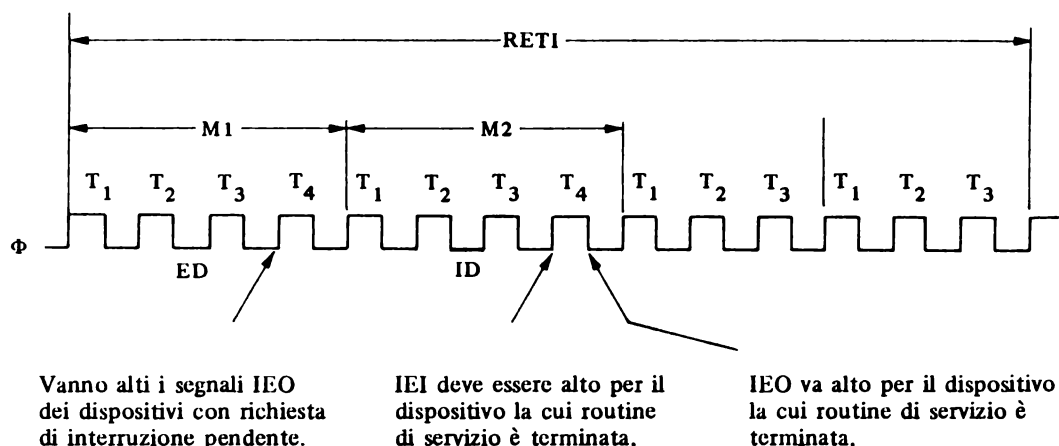



Fig. 4.19
Attivazione e disattivazione dei segnali IEI ed IEO durante l'esecuzione dell'istruzione RETI.

Alla fine della istruzione RETI relativa al dispositivo C sia A che B sono in grado di porre a livello logico basso IEO. Quando la CPU accetterà di servire la richiesta solo A si trova con IEI alto ed IEO basso e quindi solo tale dispositivo è in grado di inviare il vettore delle interruzioni: allora è mandata in esecuzione la routine relativa al dispositivo A, come d'altra parte doveva essere, visto che esso è ad un livello superiore di priorità.
Si possono verificare delle situazioni, come ad esempio quella relativa alla sequenza di istruzioni:

```
SET     5,L     CBH
                EDH
LD      C,L     4DH
```

in cui si presentano in successione i codici operativi EDH, 4DH, pur non essendo essi dovuti ad una istruzione RETI. Per evitare di interpretare erroneamente questo, ed altri casi che si possono presentare, sono stati predisposti i due FF E e G di fig. 4.17 ed è stata opportunamente organizzata la memoria ROM che serve per la decodifica di quanto presente sul bus dei dati.
Da quanto è stato esaminato si può dedurre che la risoluzione delle priorità avviene, durante il ciclo di accettazione, da quando la CPU attiva M1/ fino a quando attiva anche IORQ/. Questo è il massimo intervallo di tempo concesso perché si possano propagare nella catena i vari segnali IEI e IEO: poiché ogni dispositivo introduce un certo ritardo alla propagazione di tali segnali, esiste un limite al numero massimo di dispositivi collegabili alla catena direttamente: in pratica non possono essere presenti più di quattro dispositivi a meno di non utilizzare dell'hardware aggiuntivo.
Onde evitare che durante l'intervallo in cui i segnali di daisy-chain si propagano, qualche dispositivo possa farli variare, ogni componente della catena stessa è inibito a far variare i segnali IEI, IEO e INT/ mentre è attivo il segnale M1/.
Questa caratteristica non è rappresentata nella fig. 4.17.

4.15. Interrupt controller

L'utilizzazione dei componenti della famiglia dello Z80 semplifica notevolmente la gestione delle interruzioni, dato che i singoli componenti sono predisposti a funzionare secondo le modalità stabilite per l'unità centrale. Per altri tipi di microprocessori invece occorre utilizzare dell'hardware per la gestione delle interruzioni: allo scopo sono stati realizzati dei circuiti integrati, previsti di solito per il funzionamento con un particolare microprocessore, ma che con semplici adattamenti possono essere utilizzati con una certa generalità.

LOGIC DIAGRAM

PIN CONFIGURATION

PIN NAMES

INPUTS	
$\overline{R_0}$-$\overline{R_7}$	REQUEST LEVELS (R_7 HIGHEST PRIORITY)
$\overline{B_0}$-$\overline{B_2}$	CURRENT STATUS
$\overline{SGS}$	STATUS GROUP SELECT
$\overline{ECS}$	ENABLE CURRENT STATUS
INTE	INTERRUPT ENABLE
$\overline{CLK}$	CLOCK (INT F-F)
$\overline{ELR}$	ENABLE LEVEL READ
ETLG	ENABLE THIS LEVEL GROUP
OUTPUTS:	
$\overline{A_0}$-$\overline{A_2}$	REQUEST LEVELS (OPEN COLLECTOR)
$\overline{INT}$	INTERRUPT (ACT. LOW) (OPEN COLLECTOR)
ENLG	ENABLE NEXT LEVEL GROUP

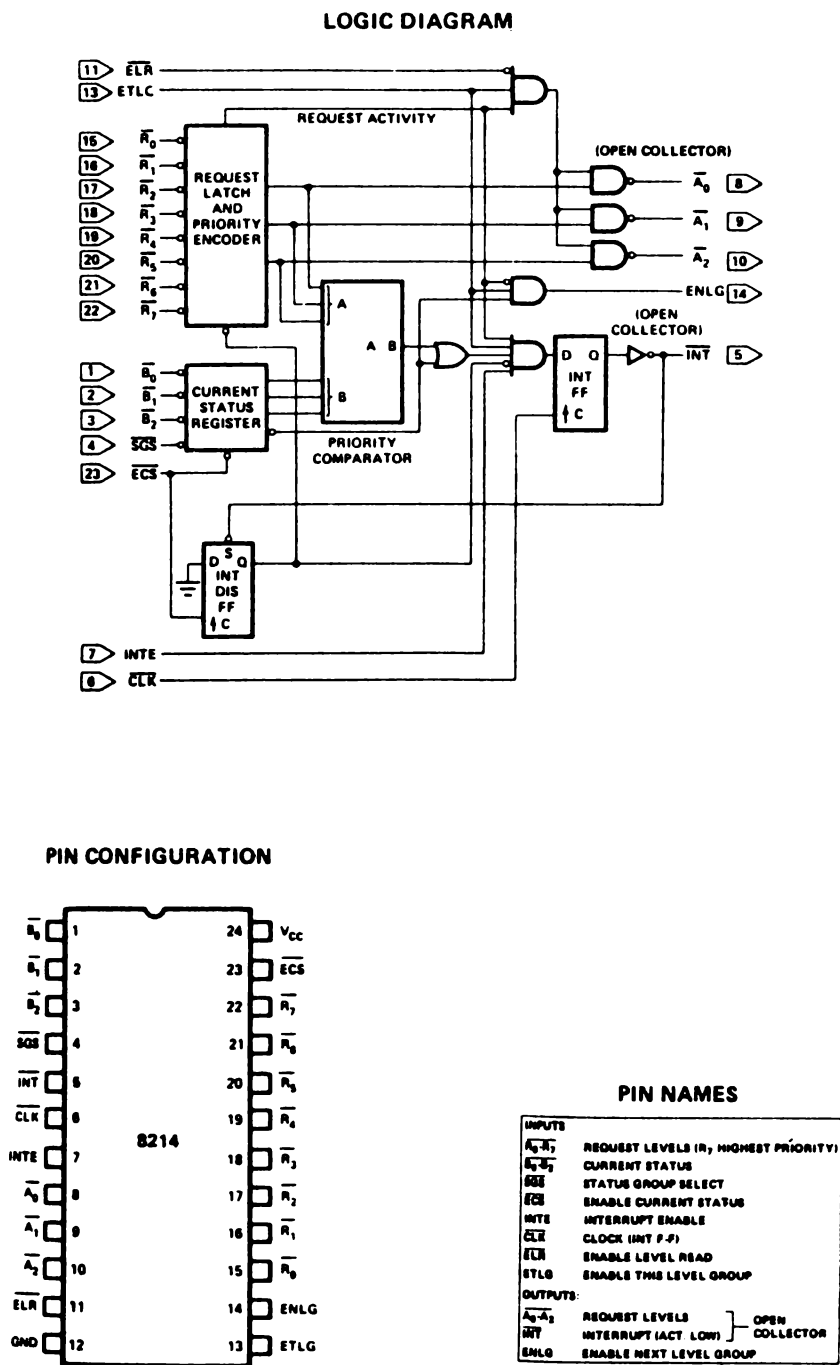



Fig. 4.20
Unità di controllo delle interruzioni 8214.
(Intel - Component Data - Catalog 1979)

Essi in genere sono adatti a funzionare secondo il modo 0 dello Z80: il processore, nella fase di accettazione delle interruzioni preleva dal bus dei dati un codice operativo particolare (RSTn) e provvede a salvare il program counter per il successivo ripristino del programma interrotto.
I controllori di interruzioni però sono in grado di fornire non solo delle istruzioni da un byte ma anche da più byte: è allora possibile fornire al processore una istruzione di CALL cioè di chiamata alla routine di servizio.
In genere viene in un primo tempo fornito il codice operativo di una istruzione di CALL, riconosciuto il quale il processore è in grado di procedere alla successiva fase di acquisizione dell'indirizzo.
La programmabilità di questi controllori consente di fare iniziare in qualsiasi locazione di memoria la routine di servizio: è possibile anche una gestione automatica delle priorità addirittura con una priorità circolare nel senso che una volta servito un dispositivo la sua priorità passa al livello più basso.
Si possono ricordare ad esempio il controllore 8259 della Intel adatto ad essere usato in particolare con i processori 8080, 8085 e 8086.
Questi controllori di interruzioni programmabili permettono in generale di collegare fino ad 8 periferiche e quindi gestire 8 diverse sorgenti di interruzioni; è spesso prevista anche la possibilità del collegamento in cascata per cui aumenta il numero di dispositivi che possono essere gestiti.
Ad esempio l'unità di controllo delle priorità delle richieste di interruzioni, l'8214, il cui schema logico è indicato in fig. 4.20, è un dispositivo il quale può gestire fino a 8 richieste distinte. Esso è in grado di determinare il livello di priorità delle richieste attive in ogni istante e di inviare una richiesta di interruzione alla CPU solo se è presente una richiesta a livello superiore a quello contenuto in un registro di stato; su quest'ultimo può essere scritto dalla CPU il livello di priorità delle richieste che essa vuol gestire.
L'8214, quando è attivo il segnale di accettazione della interruzione, è in grado di inviare quella parte del codice operativo di una istruzione di RSTn che è sufficiente per identificare quale ingresso, e quindi quale periferica, richiede la interruzione.

Capitolo 5

DISPOSITIVI DI INGRESSO E DI USCITA

5.1. Organizzazione delle interfaccie di ingresso e di uscita

I dispositivi di interfaccia, o porte, di ingresso e di uscita, consentono il colloquio fra il microprocessore ed il mondo esterno.
Tale scambio di informazioni si attua in modo simile a quello che interessa i dispositivi di memoria, ma presenta delle differenze dovute alla grande varietà di dispositivi periferici (trasduttori, attuatori, stampanti, dischi, ecc.) che possono essere presenti e che impongono, tra l'altro, delle velocità di trasferimento delle informazioni variabili entro un ampio campo e delle modalità di colloquio distinte da caso a caso.
Mentre infatti la velocità con cui avviene uno scambio di informazioni tra il microprocessore ed un dispositivo di memoria interno al microelaboratore è costante, nel caso tale scambio intercorra con dispositivi periferici la velocità può raggiungere i 250.000 bit/s, se si tratta di un'unità a dischi, oppure scendere a qualche centinaio di bit/s nel caso del trasferimento di dati con una telescrivente.
Per quanto riguarda le modalità con cui avviene lo scambio di informazioni con dispositivi esterni, anche qui ci si trova in una situazione che può essere completamente diversa da quella relativa alla memoria: per quest'ultima infatti le operazioni di lettura e/o di scrittura avvengono in modo automatico nel senso che il microprocessore non abbisogna in genere di sapere se il dispositivo di memoria è in grado, o meno, di partecipare allo scambio, dato che gli eventuali stati di attesa, quando sono presenti memorie ad elevato tempo di accesso, sono perfettamente trasparenti per quello che riguarda il microprocessore.
Nel caso invece di un colloquio con l'esterno, nella maggior parte dei casi il microprocessore deve in qualche modo testare la disponibilità della periferica allo scambio di informazioni.
Per i due motivi accennati è allora di solito presente, nel bus di controllo del microprocessore, un certo numero di linee dedicate esclusivamente al colloquio con le porte di ingresso e di uscita; tali linee entrano in attività solo quando sono eseguite quelle particolari istruzioni che riguardano le

porte di ingresso o di uscita, e fanno sì che si possa interpretare quanto presente nei bus dei dati e degli indirizzi come relativo ad una operazione con periferiche invece che con dispositivi di memoria (fig. 5.1).

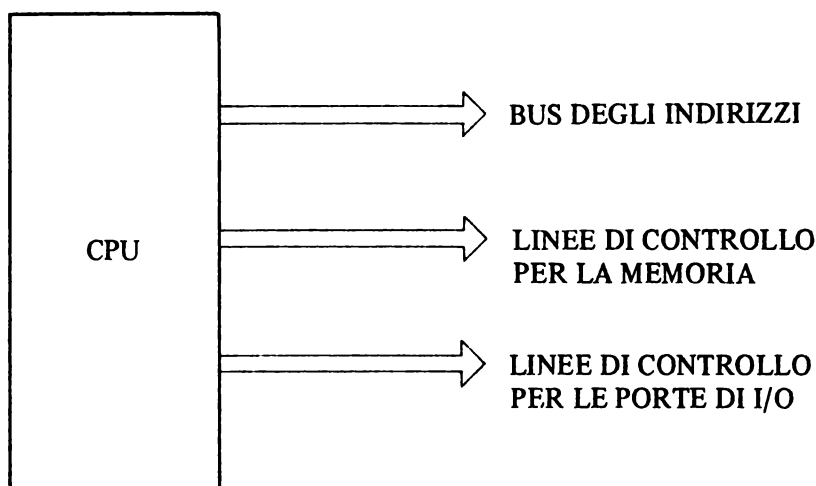



Fig. 5.1
Distinzione delle linee del bus di controllo.

Esistono due modi diversi per collegare l'unità centrale con gli organi di ingresso e uscita e per controllare il trasferimento delle informazioni: questi modi prendono il nome di I/O isolati e memory mapped I/O.

5.2. I/O isolati

Questa tecnica è utilizzabile in tutti quei microprocessori che abbiano dei segnali di controllo separati per distinguere se si tratta di operazioni di I/O o di memoria, ed inoltre che posseggano un gruppo di istruzioni a ciò dedicate. Un possibile diagramma temporale di un ciclo macchina per una operazione di scrittura su una porta di uscita è indicato nella fig. 5.2. Il segnale IORQ/ indica che gli indirizzi in quel momento presenti sul bus omonimo si riferiscono a delle porte di ingresso o di uscita. Il segnale WR/ indica che il microprocessore vuole effettuare l'operazione di scrittura. Per attivare una tale operazione sono a disposizione delle istruzioni del tipo:

OUT DEVICE

mediante le quali quanto è presente in un registro prefissato, ad esempio l'accumulatore, è inviato alla porta di uscita di indirizzo simbolico DEVICE.

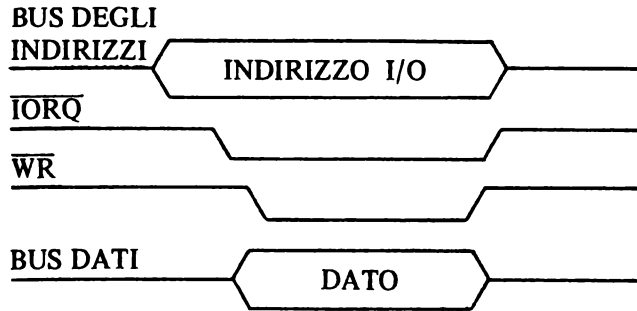



Fig. 5.2
Ciclo macchina per la scrittura di un dato su di una porta di uscita.

Analoga situazione si presenta nel caso di una lettura da porte di ingresso. Sfruttando poi il contenuto del bus degli indirizzi la porta interessata può essere selezionata con tecniche identiche a quelle viste per la decodifica degli indirizzi nel caso riguardante le memorie. Un possibile schema a blocchi per la sola decodifica degli indirizzi potrebbe essere quello di fig. 5.3.

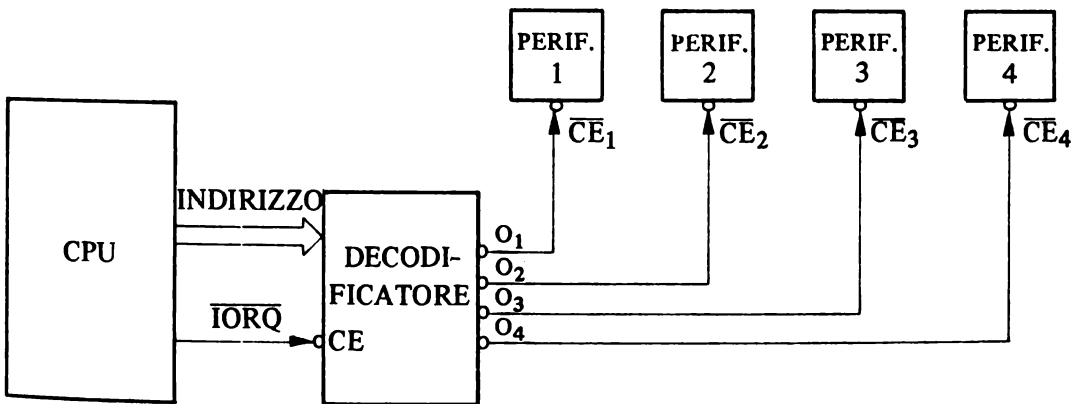



Fig. 5.3
Decodifica degli indirizzi per porte di I/O isolate.

dove si può notare come il decodificatore sia abilitato dal segnale di controllo IORQ/ che individua, se presente, una operazione relativa ad una porta di I/O.
E' evidente che per poter utilizzare la tecnica degli I/O isolati è necessario che il microprocessore abbia delle istruzioni apposite per le operazioni riguardanti le porte di I/O.
Tale tecnica presenta degli inconvenienti: in genere gli indirizzi a disposizione per le operazioni riguardanti le porte sono notevolmente inferiori a quelle a disposizione per la memoria; ad esempio in molti processori per la memoria si hanno 16 bit di indirizzo mentre per l'I/O solamente 8.
Questo comporta che solamente 256 dispositivi possono essere collegati direttamente all'unità centrale; nella pratica corrente tale limitazione è di poco peso in quanto difficilmente si raggiungono questi valori come numero di dispositivi collegati al bus.
Esiste però un inconveniente più grave dovuto al limitato numero di istruzioni in genere dedicate all'ingresso e uscita. Non è raro il caso di avere a disposizione solamente due istruzioni, IN DEVICE OUT DEVICE, con le quali si impone il trasferimento dei dati fra le porte di I/O ed uno solo dei registri interni della CPU; non è così possibile sfruttare tutti i vari modi di indirizzamento di solito a disposizione, i quali permettono di semplificare il software.
Per ovviare a tale inconveniente è possibile organizzare il trasferimento delle informazioni con le porte di I/O adottando le stesse modalità di trasferimento con la memoria se si accettano alcune restrizioni, peraltro abbastanza lievi.
In tal modo si sfrutta il maggiore numero di istruzioni che sono in genere a disposizione per il colloquio con la memoria.
Tale tecnica prende il nome di "memory mapped I/O".

5.3. Memory mapped I/O

Per illustrare tale tecnica si supponga di voler realizzare un elaboratore con un microprocessore capace di fornire 16 bit di indirizzo e che la memoria richiesta sicuramente non superi i 32K, allocati a partire dall'indirizzo 0000H. Poiché per la individuazione di una posizione su 32K sono sufficienti 15 bit, il sedicesimo bit del bus degli indirizzi non è mai utilizzato, oppure se si adotta una codifica completa, il suo valore è sempre zero. Si può allora sfruttare questa possibilità per organizzare il circuito di decodifica come in fig. 5.4.
I due decodificatori sono attivati selettivamente in base al valore assunto dal bit A15; quando esso vale 0 entra in funzione il decodificatore delle memorie, altrimenti quello relativo alle porte di ingresso e di uscita: la "mappa degli indirizzi" corrispondente è indicata nella fig. 5.5.

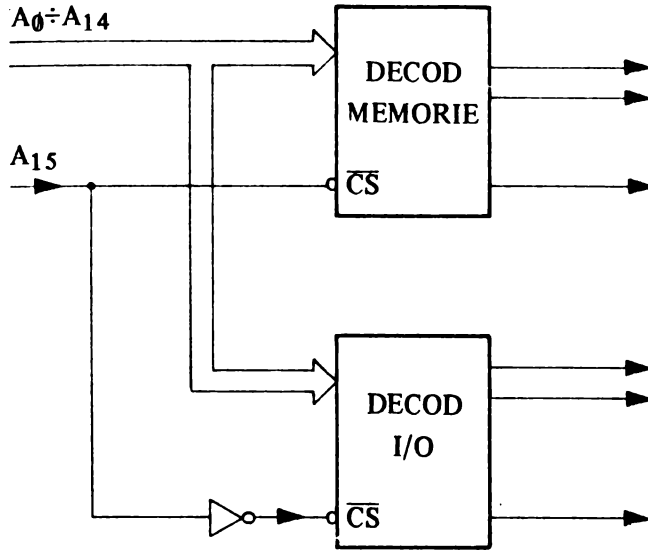



Fig. 5.4.
Decodifica degli indirizzi per memory-mapped I/O.

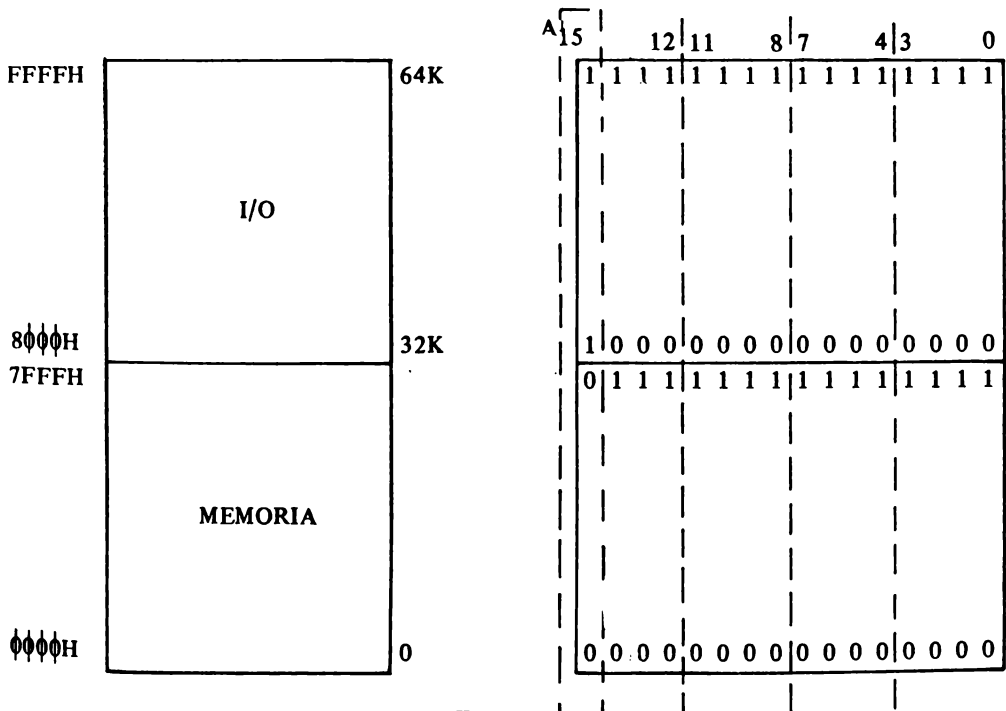



Fig. 5.5.
Mappa degli indirizzi relativa alla fig. 5.4

L'unità centrale può allora eseguire un'operazione riguardante l'I/O come una normale operazione di memoria nel campo di indirizzi da 32K a 64K: si possono così sfruttare anche per le periferiche tutte le istruzioni relative al trasferimento dati con la memoria.
Lo schema di fig. 5.4 presenta degli svantaggi: la memoria disponibile è ridotta esattamente alla metà. Si può diminuire il numero di indirizzi di memoria dedicati alle porte di I/O ricorrendo ad una decodifica che prenda in esame un numero maggiore di bit.
Si supponga ad esempio che per la memoria del microelaboratore siano ora necessari 48K: si può allora adottare lo schema di fig. 5.6, in cui è anche riportata la relativa mappa degli indirizzi.

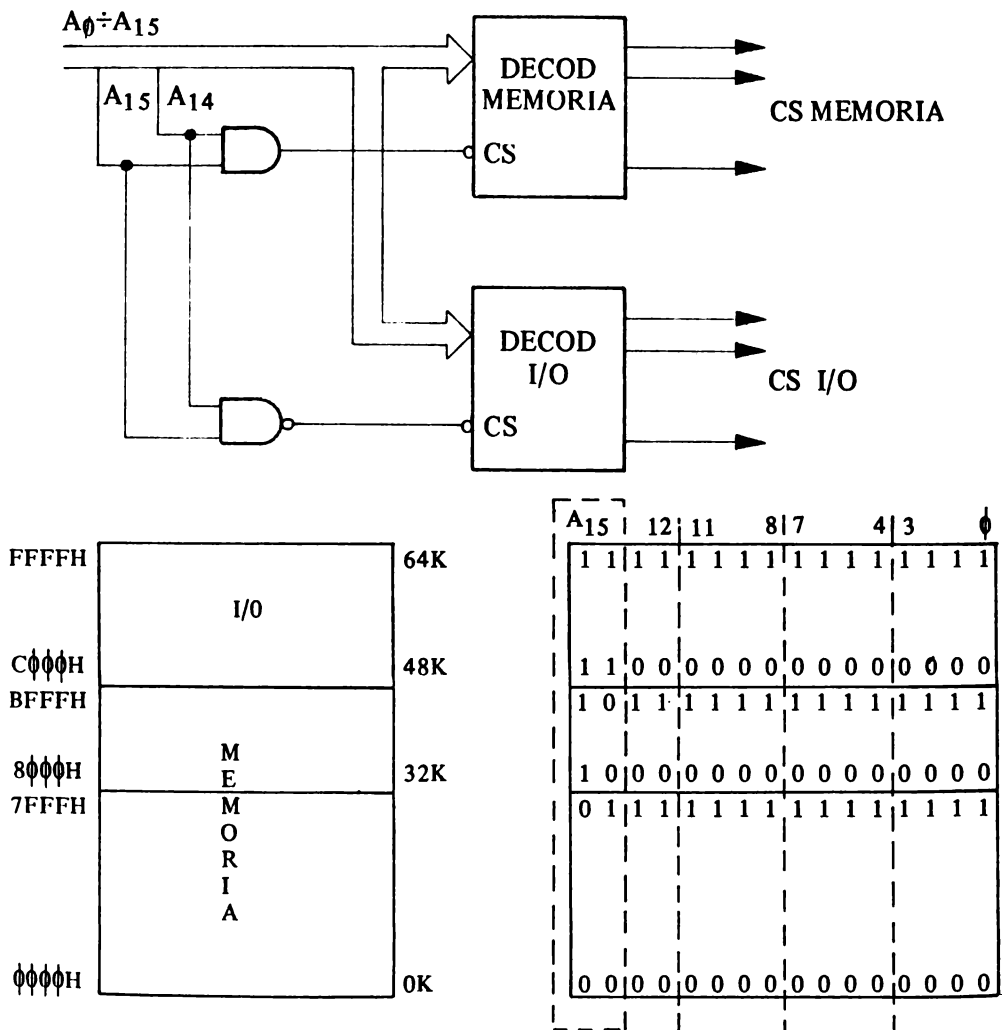



Fig. 5.6
Decodifica degli indirizzi con 16K di memory-mapped I/O.

Rispetto al caso precedente si è complicata la rete per la decodifica degli indirizzi ma l'area di memoria dedicata ai dispositivi di I/O si è ridotta a soli 16K; è ovvio che essa può essere ulteriormente ridotta utilizzando opportune reti combinatorie per l'abilitazione dei decodificatori.
Si noti inoltre che non sempre è necessario ricorrere a due decodificatori separati come negli schemi precedenti; ad esempio in un microelaboratore in cui sono necessari solo 32K di memoria organizzati in banchi da 8K ciascuno e nel quale servano al massimo 4 porte di ingresso/uscita si può adottare lo schema di fig. 5.7:

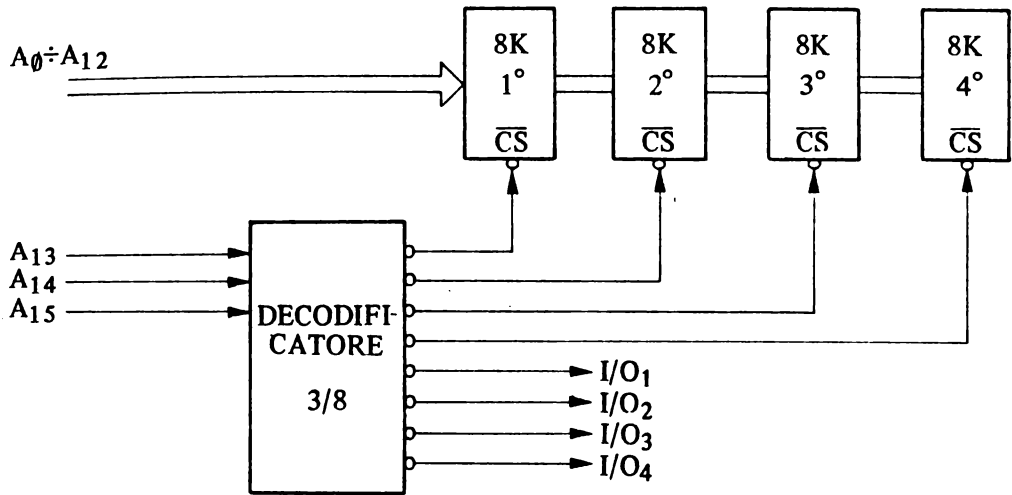



Fig. 5.7
Decodifica di indirizzi per 4 memory-mapped I/O.

Per individuare uno dei quattro dispositivi si può utilizzare ora uno qualsiasi degli indirizzi che si trovano compresi entro il rispettivo campo di 4K: ad esempio gli indirizzi 8000H, 8001H, 8002H, ecc. individuano tutti lo stesso dispositivo indicato in figura con I/O 1.

5.4. Confronto fra le tecniche di I/O isolati e Memory-Mapped I/O

La tecnica dell'I/O isolato lascia a disposizione l'intero campo di memoria indirizzabile da parte della CPU: in qualche caso occorre utilizzare tale tecnica perché c'è la necessità di avere tutta la memoria a disposizione.
In genere il circuito di interfaccia con gli organi periferici può risultare più semplice dato che si utilizzano segnali già forniti all'uopo dal microprocessore.
Un altro vantaggio consiste nel fatto che la lettura del programma sorgente in simbolico è molto più semplice in quanto vengono usate delle istruzioni facilmente distinguibili da quelle impiegate per il trasferimento in memoria. Nel caso invece di memory mapped I/O l'unica informazione che permette di distinguere che una operazione con memoria si riferisce effettivamente a questa oppure ad un organo di ingresso-uscita è l'analisi degli indirizzi, sicuramente meno agevole rispetto ad un codice simbolico distinto.
Inoltre il sistema di decodifica, molto semplice negli esempi presentati, può invece diventare complesso qualora non si possa concedere troppo spazio di memoria agli organi periferici per cui è necessario prendere in esame un discreto numero di bit per fare la distinzione fra memoria oppure I/O.
Esistono inoltre dei circuiti periferici dove tale tecnica è difficilmente applicabile in quanto sono stati previsti per utilizzare i segnali relativi alle istruzioni di IN e di OUT. E' questo ad esempio il caso dei componenti della famiglia dello Z80.

5.5. Segnali di controllo per la comunicazione con gli organi di ingresso/uscita

Poiché le porte di ingresso/uscita devono essere collegate al bus di sistema, esse devono possedere quelle caratteristiche che questa organizzazione impone per un corretto funzionamento: in particolare le porte di ingresso devono poter assumere anche lo stato di alta impedenza oppure essere del tipo open collector; quelle di uscita devono invece presentare una impedenza di ingresso vista dal bus, la più elevata possibile.
Si fa presente che nel trasferimento dei dati ci si riferisce sempre implicitamente all'unità centrale per stabilire il verso: una porta di ingresso è quella che trasferisce dei dati dalla periferia verso la CPU; viceversa accade per una di uscita.

Uno schema di principio utilizzabile per il collegamento di porte sia di ingresso che di uscita è riportato nella fig. 5.8.

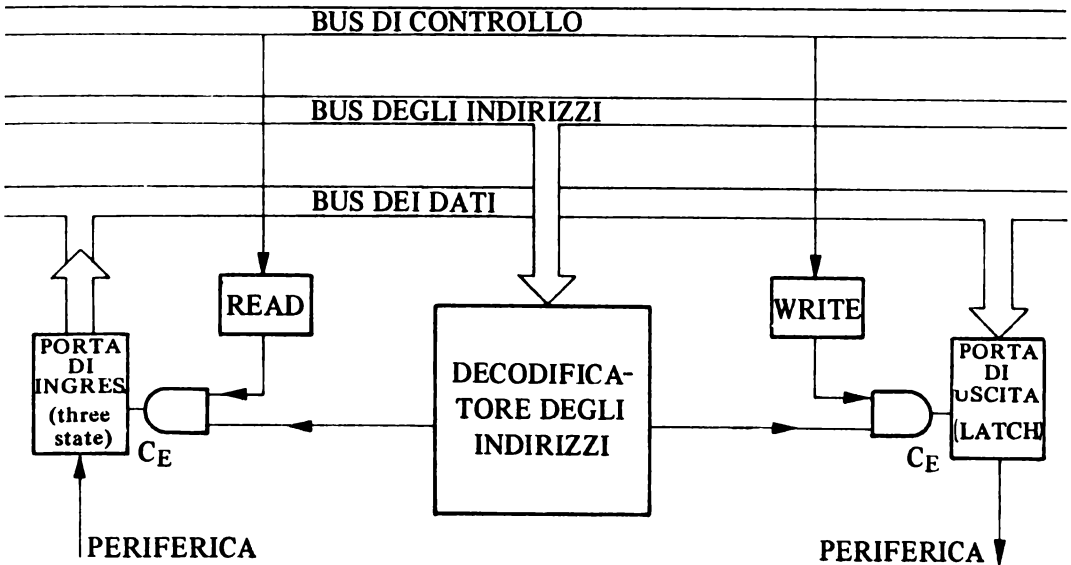



Fig. 5.8
Schema di principio per l'utilizzazione delle porte di I/O.

La porta di uscita deve essere in grado di memorizzare i dati inviati dal microprocessore, cioè deve essere di tipo latch, in quanto in caso contrario la periferica dovrebbe prelevare il dato "al volo" cioè in quel breve intervallo di tempo nel quale l'unità centrale lo mette a disposizione sul bus dei dati.
L'abilitazione delle due porte, indicata in fig. 5.8 solo in forma simbolica, è attuata per il tramite di una decodifica degli indirizzi, ed utilizza i segnali di controllo, READ o WRITE, per individuare l'operazione che si deve eseguire.
Queste indicazioni possono essere dedotte in modo diverso a seconda che si utilizzi la tecnica dell'I/O isolato o del memory mapped I/O. Analoga considerazione va fatta nei confronti degli indirizzi da utilizzare nei due casi.
Si noti che non è prevista in fig. 5.8 alcuna segnalazione per informare

sia che un dato è disponibile per la periferica, o per la CPU, sia che la periferica è pronta a ricevere, o inviare, un nuovo dato.
Ciò costituisce un notevole inconveniente da un punto di vista applicativo tanto che lo schema di fig. 5.8 può essere utilizzato solamente in casi molto semplici e particolari.
Si consideri ad esempio il caso di dover inviare dei dati ad una normale telescrivente: quando il microprocessore invia il primo dato a tale periferica, esso effettua una operazione che non differisce concettualmente da una normale scrittura su una posizione di memoria.
Una notevole differenza si nota invece quando il microprocessore deve inviare il dato successivo: infatti una volta accettato un dato la telescrivente non è più in grado di riceverne altri finché non lo ha stampato, e a ciò impegna un certo intervallo di tempo, che è notevolmente lungo confrontato con i normali tempi di funzionamento della unità centrale. Questo problema non si presenta nel caso di una memoria: infatti appena terminata l'operazione di scrittura è possibile eseguirne un'altra in successione, con la certezza che si presentano le stesse situazioni della scrittura precedente.
Analoga situazione si presenza ovviamente nel caso di ingresso da una periferica; si pensi ad esempio ad una tastiera: una volta acquisito un dato la CPU deve essere informata quando ne è disponibile un altro per una nuova lettura.
Lo schema proposto può quindi essere utilizzato solamente in casi particolari, quando, cioè, la periferica è in ogni istante disponibile per un trasferimento di informazioni.
Si pensi ad esempio ad un trasduttore di una grandezza fisica lentamente variabile nel tempo, ad esempio un termometro per la misura della temperatura ambiente: in questo caso il dato può essere considerato sempre presente in ingresso, date le caratteristiche del trasduttore.
In altri casi, quando cioè, una volta avvenuta una determinata operazione di lettura o di scrittura, la periferica per un certo periodo di tempo è occupata e non può quindi accettare o inviare un nuovo dato, sono necessarie al microprocessore ulteriori informazioni per poter effettuare il colloquio con il dispositivo interessato: tali informazioni sono contenute in opportuni segnali detti di handshaking.
Uno schema di principio, relativo al collegamento con una periferica che invia dati al microelaboratore, è indicato in fig. 5.9: in esso sono indicati i due segnali di handshaking "dato pronto" e "dato accettato", gestiti rispettivamente dalla periferica e dal microprocessore.
Il colloquio avviene nel modo seguente: quando la periferica ha pronto un dato lo presenta agli ingressi della porta di interfaccia ed invia contemporaneamente la segnalazione di "dato pronto".

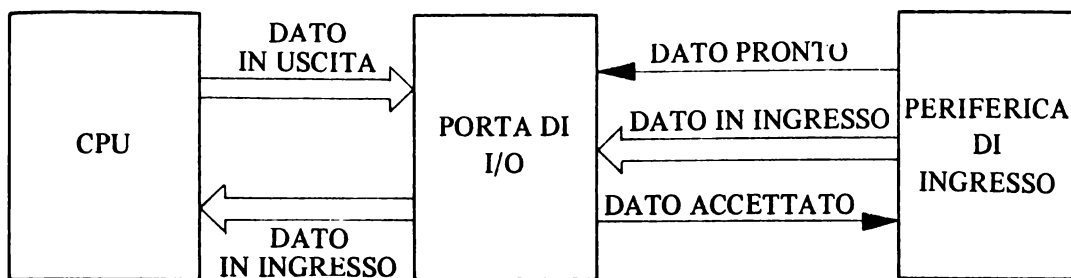



Fig. 5.9
Collegamento di una periferica di ingresso con segnali di handshaking.

La CPU, quando controlla questo segnale e lo trova attivo, legge il dato dalla porta di ingresso. Una volta effettuata tale lettura, la CPU a sua volta invia alla porta d'uscita la segnalazione di "dato accettato" per informare la periferica che il dato è stato acquisito.
A seguito di tale segnalazione la periferica può procedere all'invio di un nuovo dato e così via.
Nello schema di fig. 5.9 si può notare che i segnali di handshaking devono essere gestiti sia dalla CPU che dalla periferica; in pratica ciò può essere effettuato in modo automatico con dell'hardware a ciò predisposto: ad esempio si potrebbero organizzare le cose in modo tale che la segnalazione di "dato pronto" automaticamente disattivi quella di "dato accettato".
In tal modo si è certi di acquisire una sola volta il dato inviato dalla periferica, in quanto, così facendo, è possibile sincronizzare il funzionamento della periferica con quello dell'unità centrale.
Di solito si verifica che è la periferica ad avere una velocità di funzionamento più bassa, cui il microprocessore deve adeguarsi, con gli inconvenienti analizzati nel descrivere la tecnica del polling.
Un'alternativa al circuito di fig. 5.9 consiste nell'utilizzare le richieste di interruzione una volta che il dato all'ingresso è pronto: un possibile schema potrebbe essere quello riportato nella fig. 5.10.
La periferica trasmette il dato da inviare ad un buffer latch che lo memorizza su comando, fornito dalla stessa periferica mediante il segnale di "dato pronto"; quest'ultimo provoca anche l'attivazione del segnale per la richiesta di interruzione.
Durante l'esecuzione della routine di servizio la CPU procede alla lettura del dato disponibile alla porta di ingresso. Il comando di lettura disattiva la richiesta di interruzione e contemporaneamente attiva il segnale di "dato

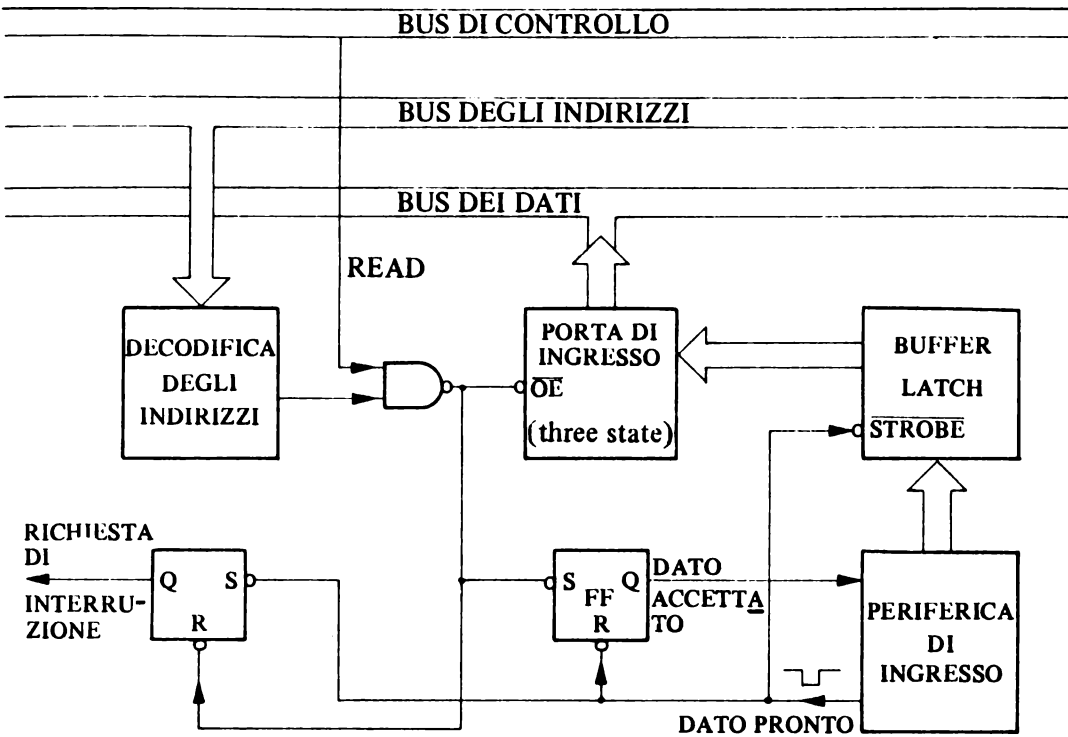



Fig. 5.10
Schema di handshaking con richiesta di interruzione per una periferica di ingresso.

accettato" per cui la periferica può procedere ad una successiva trasmissione di un dato verso la CPU. Si noti che l'attivazione di "dato pronto" resetta in ogni caso il FF che ha fornito l'indicazione di "dato accettato". Nel caso invece di scrittura di un dato su una periferica si deve controllare se la periferica a cui è destinato è pronta a riceverlo. Un possibile schema di principio è riportato nella fig. 5.11.
La periferica deve inviare un segnale di "periferica pronta" per indicare che è disponibile a ricevere un nuovo dato. La CPU, constatata la disponibilità della periferica al colloquio, invia il dato in una porta di uscita in grado di memorizzarlo; informa quindi la periferica che il dato è disponibile attivando il segnale di "dato pronto". All'arrivo di questa segnalazione

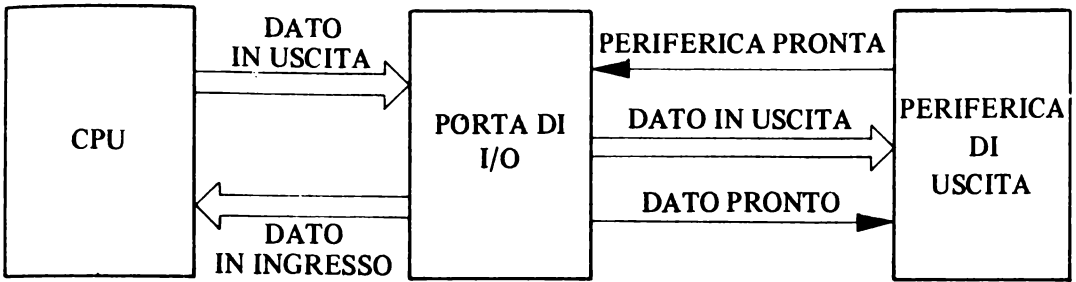



Fig. 5.11
Collegamento di una periferica di uscita con segnali di handshaking.

la periferica preleva il dato presente sulla porta di uscita ed informa quindi la CPU di essere nuovamente pronta per ricevere un nuovo dato. Tale informazione è interpretata dalla CPU nel senso che il dato precedentemente inviato è stato ricevuto, per cui le è possibile inviare un nuovo dato e così via.
Anche in questo caso si presenta l'inconveniente che la CPU può sprecare molto tempo al solo scopo di controllare se la periferica è pronta a ricevere un nuovo dato: per evitare questo si ricorre come la solito alla tecnica delle interruzioni adottando uno schema del tipo indicato nella fig. 5.12.
Quando la periferica è in grado di ricevere un dato attiva il segnale di "periferica pronta" che provoca una richiesta di interruzione; una volta mandata in esecuzione la routine di servizio la CPU invia un dato alla porta latch di uscita. Con ciò non solo si segnala alla periferica che c'è un nuovo dato a disposizione ma contemporaneamente si disattiva il segnale di richiesta di interruzione. Una volta che la periferica ha acquisito il dato, se ne abbisogna di altri, essa attiva il segnale di "periferica pronta", e ciò provoca automaticamente la disattivazione del segnale di dato pronto per cui ci si ritrova nelle condizioni di partenza.
Quanto è stato finora illustrato è relativo solo a dei trasferimenti di informazione unidirezionali; è abbastanza semplice combinare le due modalità di colloquio per ottenere un trasferimento bidirezionale.

5.6. Cicli di input e di output nello Z80

Il diagramma temporale di un ciclo riguardante una porta di ingresso o di uscita, nello Z80, è riportato nella fig. 5.13, dove sono indicati contempo-

BUS DI CONTROLLO

BUS DEGLI INDIRIZZI

BUS DEI DATI

WRITE

DECODIFICA DEGLI INDIRIZZI

PORTA DI USCITA

$\overline{CE}$

LATCH

DATO

RICHIESTA DI INTERRUZIONE

Q S R

S Q R

DATO PRONTO

PERIFERICA DI USCITA

PERIFERICA PRONTA

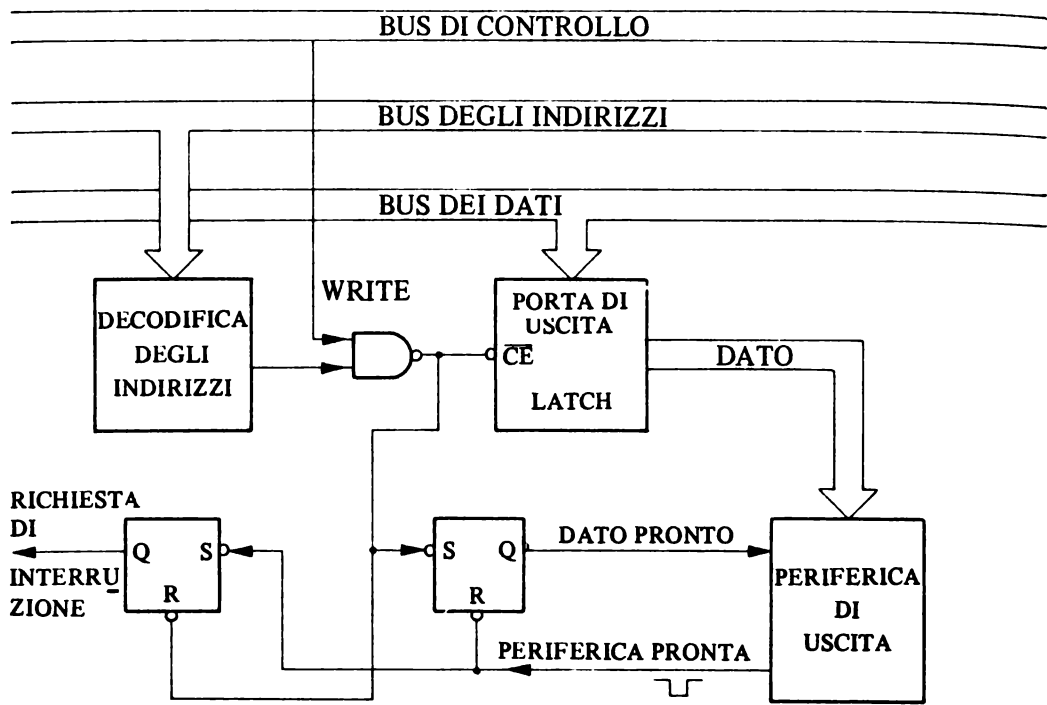



Fig. 5.12
Schema di handshaking con richiesta di interruzione per una periferica di uscita.

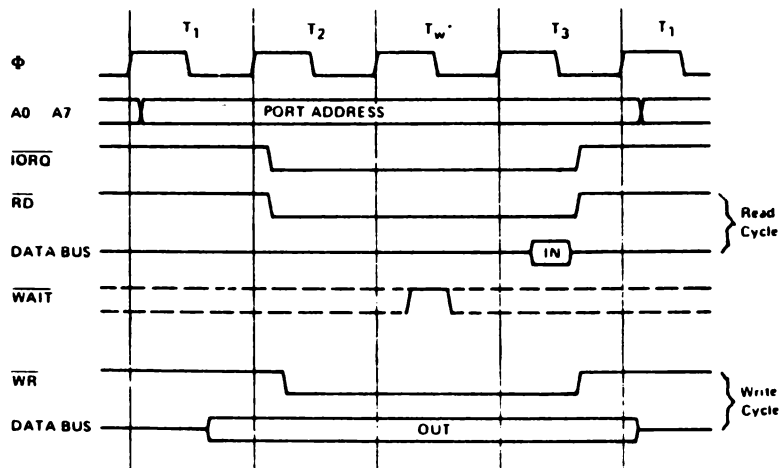



Fig. 5.13
Cicli di input e output nello Z80.
(Mostek, Z80 - Microcomputer Data Book 1981).

raneamente sia i segnali che interessano una operazione di ingresso che quelli relativi ad una di uscita.
All'inizio del primo periodo di clock T_1 è inviato sul bus degli indirizzi quello che individua la porta interessata; si noti che a ciò sono dedicati solamente i primi otto bit A0-A7 per cui al massimo sono indirizzabili 256 dispositivi. I due segnali IORQ/ ed RD/, che assieme qualificano una operazione di lettura da dispositivi di ingresso, sono attivati all'inizio del secondo periodo; con questi due segnali può essere attivata la selezione del dispositivo interessato. Sul fronte di discesa del segnale di clock in T_3 il microprocessore preleva il dato richiesto. Per aumentare il tempo a disposizione per fornire il dato da parte del dispositivo periferico è inserito dalla CPU, automaticamente, un periodo di attesa fra T_2 e T_3.
In tal modo il tempo a disposizione per una lettura è di 2,5 periodi di clock, che ovviamente vanno diminuiti dei tempi di ritardo e di set-up introdotti dal decodificatore di indirizzo, dai vari buffer e porte.
Nel caso tale tempo non sia sufficiente si possono introdurre ulteriori periodi di attesa, prima che inizi il periodo T_3 e ciò può essere ottenuto con degli schemi analoghi a quelli visti per le memorie.
Quando invece è eseguita un'operazione di scrittura, già dal primo periodo di clock T_1 sono disponibili, oltre agli indirizzi, i dati da inviare all'uscita. All'inizio del periodo T_2 sono attivati dal microprocessore IORQ/ e WR/, con ovvio significato. Anche in questo caso per aumentare il tempo a disposizione alla porta di uscita è automaticamente inserito un periodo di attesa fra T_2 e T_3, e come al solito se esso non fosse sufficiente, si può, con dell'hardware opportuno, inserirne altri.

Capitolo 6
I/O PARALLELI

6.1. Trasmissione di dati in parallelo fra CPU e periferiche

Si analizzano ora i vari circuiti di interfaccia con cui si possono mettere in comunicazione con la CPU dispositivi, esterni al microelaboratore, i quali siano in grado solo di trasmettere, o ricevere, informazioni costituite da un certo numero di bit in parallelo.
In quanto segue non si metteranno in evidenza le caratteristiche statiche e dinamiche che debbono essere proprie di tali interfaccie, dato che esse si possono desumere con considerazioni analoghe a quelle già viste per i dispositivi di memoria, ma solo gli schemi di principio che si possono adottare nelle diverse situazioni.

6.1.1. *Buffer unidirezionale*

Il più semplice caso che si può presentare è la realizzazione di una interfaccia tra la CPU ed una periferica che invia dati, cioè una periferica di ingresso; la realizzazione circuitale, indicata schematicamente in fig. 6.1, fa uso in questo esempio di un semplice buffer.

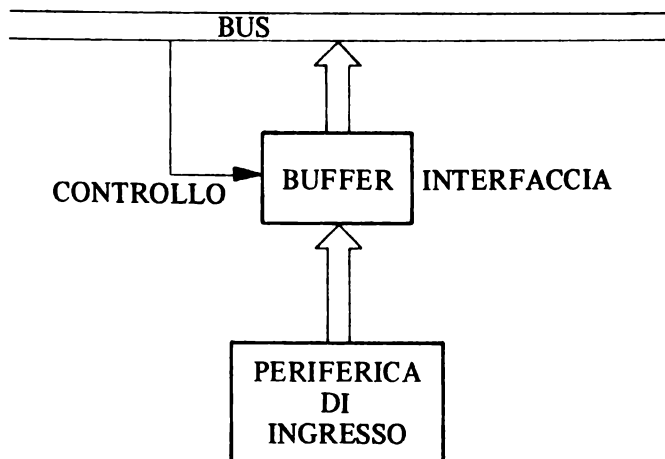



Fig. 6.1
Interfaccia di ingresso: schema a blocchi.

Un segnale del bus di controllo del sistema permette di togliere il buffer dallo stato di alta impedenza per consentire il transito dei segnali inviati dalla periferica al bus dei dati. Uno dei tanti possibili buffer utilizzabili potrebbe essere l'AmZ8144 il cui schema a blocchi è riportato nella fig. 6.2.

INPUTS		OUTPUT
OE	D_i	Y
H	X	Z
L	H	H
L	L	L

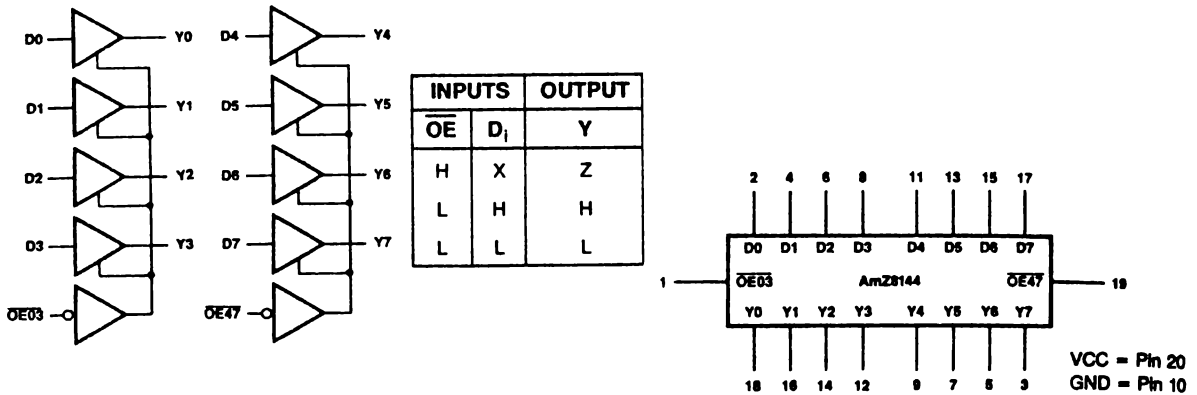



Fig. 6.2
Schema a blocchi del buffer AmZ8144 (AMD - MOS/LSI Data Book 1980).

Con tale componente è possibile immettere nel bus dati anche solo quattro bit di informazione se la periferica può emettere informazioni in questo formato. I due ingressi di abilitazione possono essere diversamente utilizzati a seconda delle necessità.
Ad esempio, con l'impiego dello Z80 come unità centrale, uno schema di principio potrebbe essere quello di fig. 6.3; in esso l'abilitazione del buffer avviene solamente quando è mandata in esecuzione una istruzione di lettura di una periferica (IORQ/ = RD/ = ∅) e quando tale operazione è indirizzata a quella particolare periferica.
Nello schema di fig. 6.3 si è supposto di utilizzare la tecnica dell'I/O isolato; qualora si voglia adottare la tecnica del memory-mapped I/O le modifiche da apportare sono estremamente semplici e consistono nell'inviare alla porta OR il segnale di controllo MREQ/ al posto di IORQ/.
Nel caso si debba collegare una periferica di uscita non è richiesta in genere al buffer la caratteristica di thre-estate ma quella di latch. In questo caso il buffer è utilizzato come separatore fra il bus ed il circuito esterno oppure come driver per aumentare il carico pilotabile rispetto a quanto sarebbe possibile con un collegamento diretto al bus.

6.1.2. *Buffer bidirezionali*

In alcuni casi la periferica abbisogna sia di inviare dati all'unità centrale sia di riceverne, però attraverso delle linee separate. Sono per tale scopo di-

sponibili dei buffer bidirezionali, come ad esempio l'8216 il cui schema logico è riportato nella fig. 6.4.

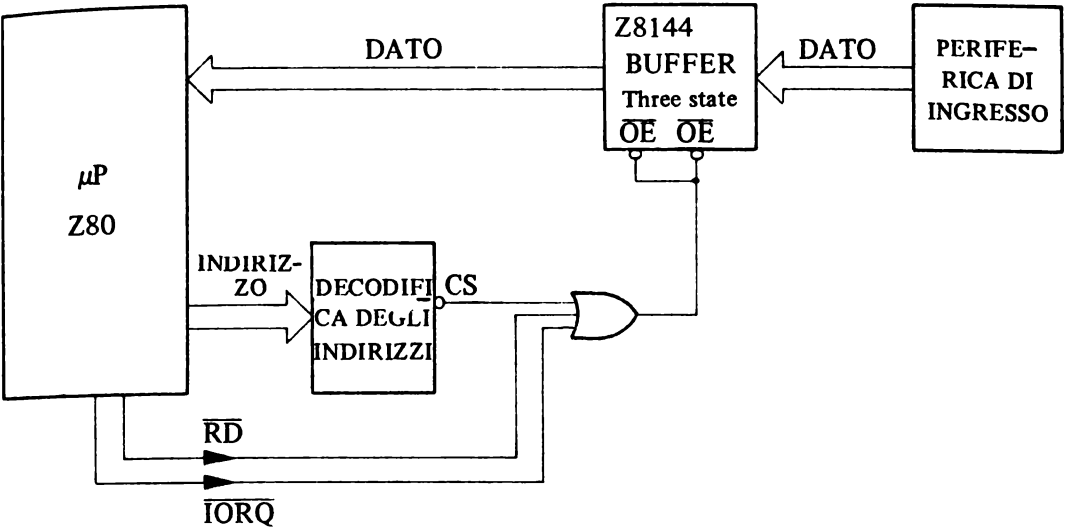



Fig. 6.3
L'AmZ8144 in un sistema con lo Z80.

PIN CONFIGURATION

LOGIC DIAGRAM
8216

PIN NAMES

DB_0-DB_3	DATA BUS BI-DIRECTIONAL
DI_0-DI_3	DATA INPUT
DO_0-DO_3	DATA OUTPUT
$\overline{DIEN}$	DATA IN ENABLE DIRECTION CONTROL
$\overline{CS}$	CHIP SELECT

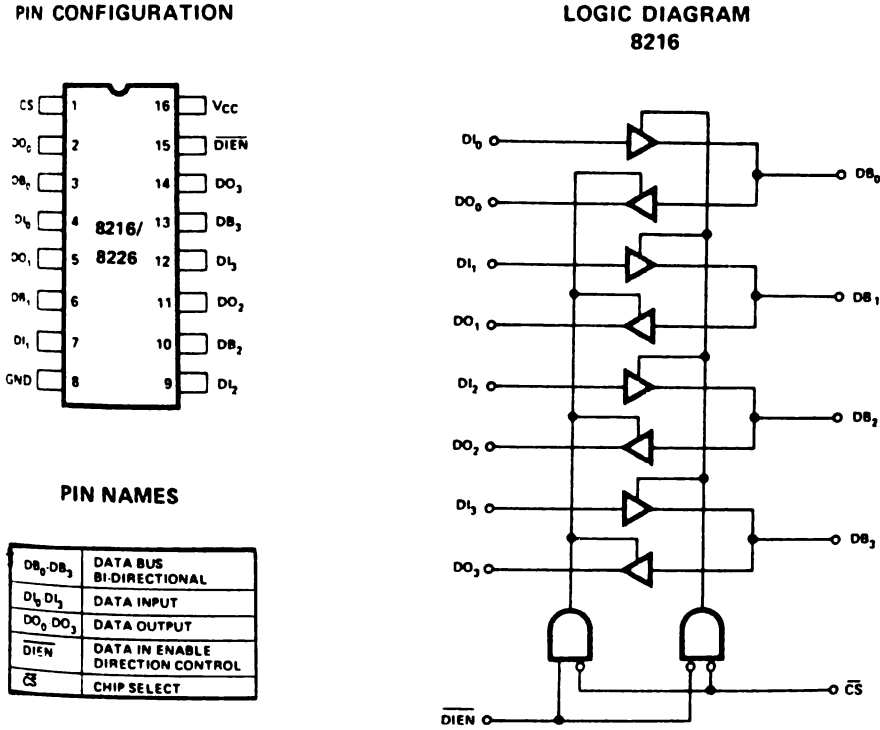



Fig. 6.4
Schema logico del buffer bidirezionale a 4 bit 8216. (Intel Component Data Catalog 1979).

L'8216 ha quattro linee, DB, collegabili direttamente al bus del sistema, attraverso le quali le informazioni possono transitare nelle due direzioni. Esistono inoltre altre otto linee che permettono il collegamento con la periferica: quattro, DO_i, servono per l'uscita, altre quattro, DI_i, per l'ingresso dei dati. Ovviamente per poter permettere la trasmissione di un byte è necessario utlizzare una coppia di tali buffer collegati in parallelo. Gli ingressi di controllo CS/ e DIEN/ consentono rispettivamente di abilitare il buffer e di stabilire la direzione con cui avviene il trasferimento della informazione; si osservi che con ingresso CS/ non attivato sia le uscite DB_i che quelle DO_i sono nello stato di alta impedenza. Riferendoci sempre allo Z80 come unità centrale l'8216 può essere collegato secondo lo schema indicato in fig. 6.5.

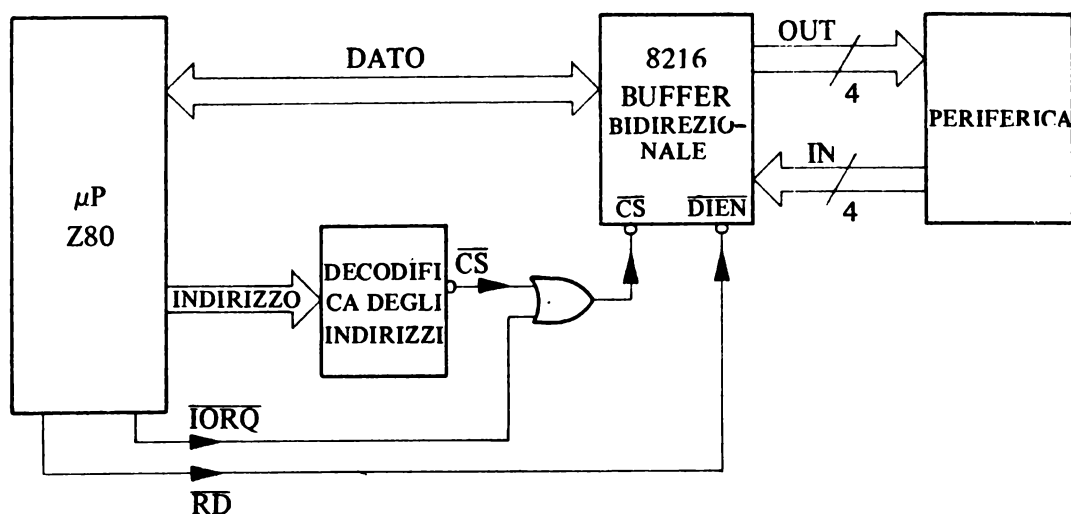



Fig. 6.5
L'8216 in un sistema con lo Z80.

Il buffer è abilitato solo se lo Z80 sta eseguendo una operazione di I/O ad un determinato indirizzo, quello che corrisponde alla periferica; con il segnale RD/ non attivo l'8216 si dispone per una scrittura di dati, cioè per l'invio di essi alla periferica. Nel caso di una lettura, tale operazione è qualificata dal segnale RD/ inviato all'ingresso DIEN/. In tal modo il buffer viene tolto dallo stato di alta impedenza nei riguardi del bus dei dati solo quando è necessario.
Ovviamente lo schema può anche essere utilizzato per un trasferimento di un byte ricorrendo a due 8216 in parallelo.
Se la periferica è di tipo bidirezionale ed utilizza per la trasmissione le stesse linee sia per l'ingresso che per l'uscita si può ancora utilizzare l'8216 collegando fra loro le linee per l'ingresso DI_i con quelle per l'uscita DO_i:

si ottiene in tal modo un buffer bidirezionale. In quest'ultimo caso è anche possibile utilizzare un buffer bidirezionale con tante linee di ingresso quante sono quelle di uscita, come ad esempio l'AmZ8104, il cui schema logico è riportato nella fig. 6.6.

VCC = Pin 20
GND = Pin 10

FUNCTIONAL TABLE

Inputs	Conditions		
Chip Disable	0	0	1
Transmit/Receive	0	1	X
A Port	Out	In	HI-Z
B Port	In	Out	HI-Z

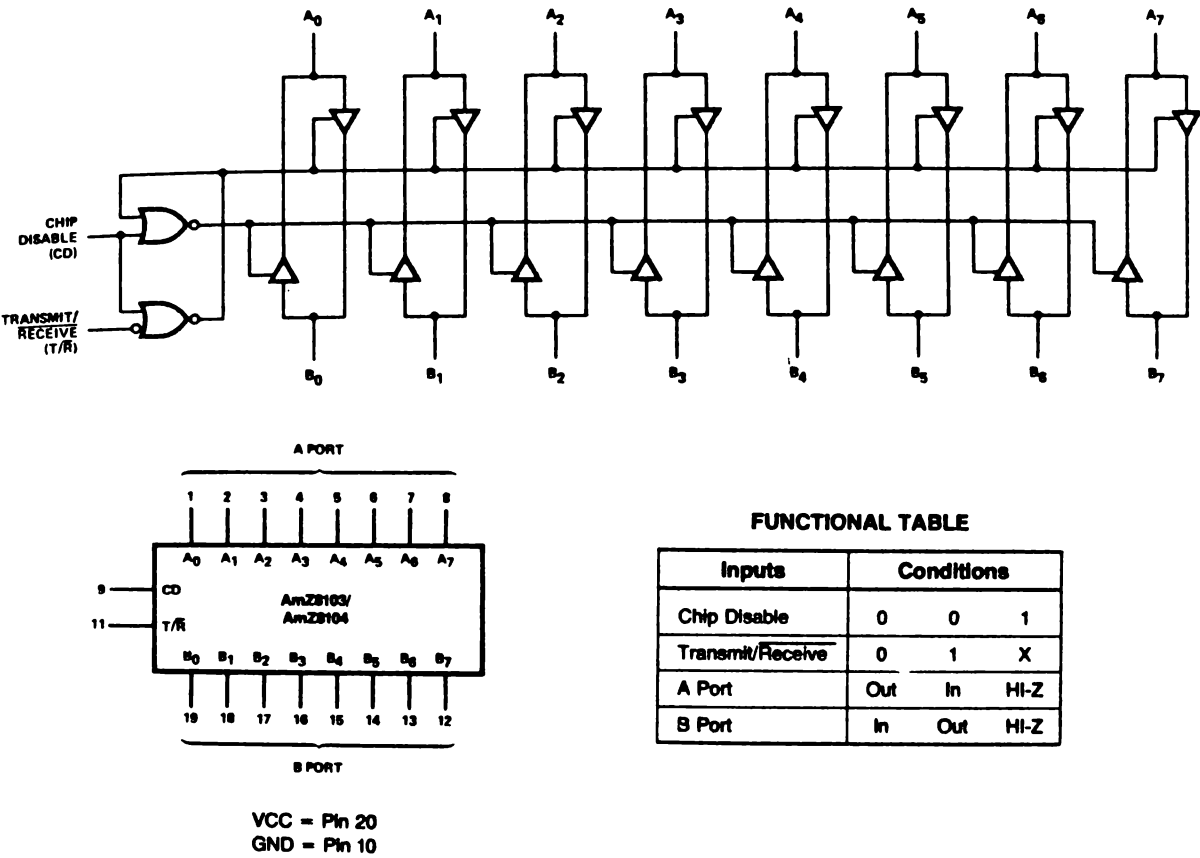



Fig. 6.6
Schema logico del buffer bidirezionale a 8 bit AmZ8104. (AMD - MOS/LSI Data Book 1980)

Esso presenta due porte da otto ingressi, individuate come porta A e porta B; sono presenti i soliti ingressi di comando (chiamati in tal caso con nomi diversi) che svolgono le stesse funzioni viste nel caso dell'8216.

6.1.3. *Buffer di tipo latch*

I due buffer presi in considerazione possono essere utilizzati solo se sono

soddisfatte le seguenti condizioni:

in fase di scrittura quando l'unità centrale invia un dato, la periferica deve essere in grado di acquisirlo in quel breve intervallo di tempo durante il quale esso è disponibile sul bus dei dati;

in fase di lettura la periferica deve essere in grado di fornire il dato quando l'unità centrale va a leggere il buffer di interfaccia.

Esistono poche periferiche che presentano tali caratteristiche e soprattutto che hanno una velocità di funzionamento compatibile con quella della CPU: in queste situazioni si possono inserire nel funzionamento dell'unità centrale alcuni stati di attesa, come visto in precedenza.
Nella maggior parte dei casi il dato inviato dall'unità centrale deve rimanere disponibile per la periferica per un tempo relativamente lungo, se confrontato con la velocità del microprocessore: non è conveniente perciò utilizzare un numero elevato di stati di attesa, in quanto si rallenta il funzionamento dell'intero sistema. Anche nei confronti dell'operazione di lettura da periferica si presentano casi analoghi: la periferica è spesso in grado di fornire un dato solo per un breve intervallo di tempo e quest'ultimo non è sincronizzato con quello in cui l'unità centrale procede all'acquisizione.
Per risolvere tali situazioni si deve ricorrere a dei buffer di tipo latch, i quali sono in grado di conservare l'informazione. Un dispositivo di questo tipo è l'AmZ8173 il cui schema è riportato in fig. 6.7.

AmZ8173

Inputs			Internal	Outputs	Function
$\overline{OE}$	G	D_i	$\overline{Q}_i$	Y_i	
H	X	X	X	Z	Hi-Z
L	H	L	H	L	Transparent
L	H	H	L	H	
L	L	X	NC	NC	Latched

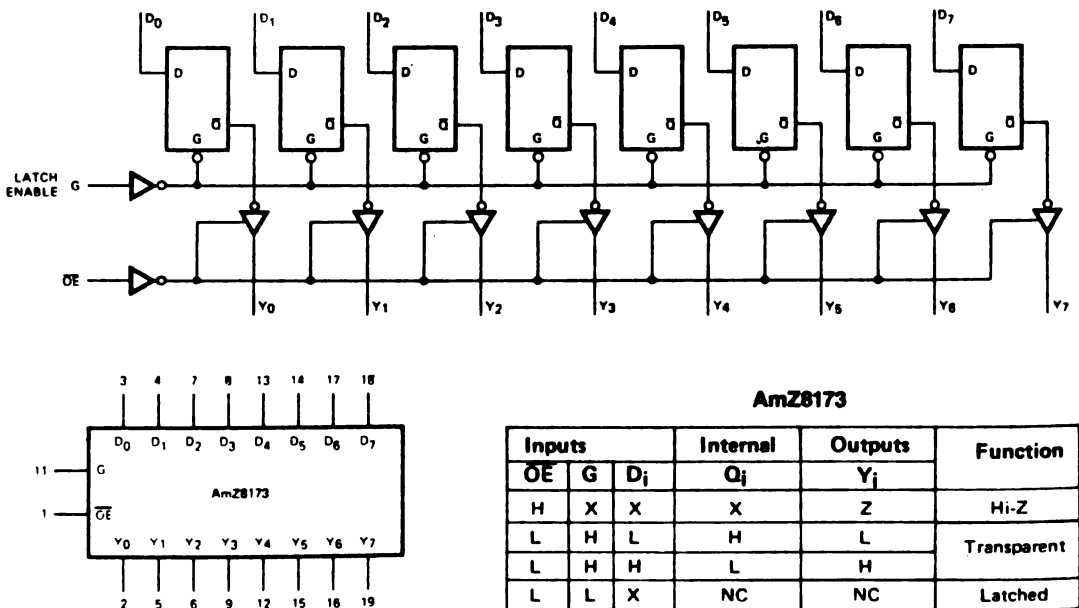



Fig. 6.7
Schema logico del buffer-latch unidirezionale a 8 bit AmZ8173 (AMD-MOS/LSI Data Book 1980).

Si tratta di un buffer unidirezionale che contiene anche 8 flip-flop (FF) in grado e di memorizzare l'informazione presente agli ingressi D_i e di renderla disponibile all'uscita Y_i con opportuni comandi.
A ciò è preposto l'ingresso OE/ (output enable) che permette di rendere disponibile all'uscita Y_i quanto è presente sui dispositivi di memoria interni: per ottenere ciò è necessario far assumere ad OE/ un valore logico basso; in caso contrario l'uscita si porta nello stato di alta impedenza.
L'altro ingresso di LACHT ENABLE, G, consente di memorizzare i dati presenti all'ingresso; come si può notare dalla tabella di verità, se tale ingresso assume un valore logico alto quanto è presente agli ingressi D_i è trasferito anche all'uscita dei FF, (e quindi all'uscita del dispositivo se anche l'ingresso OE/ è attivato).
Questo modo di funzionamento è analogo a quello dei buffer privi di elementi di memoria prima visti.
La memorizzazione dei dati si attua sul fronte di discesa del comando G, dopo di che, se G permane a livello basso, all'uscita dei FF è presente solo quanto è stato memorizzato precedentemente, qualunque sia il valore assunto degli ingressi. Un possibile schema di utilizzazione di tale componente per una periferica di uscita è quello indicato in fig. 6.8.

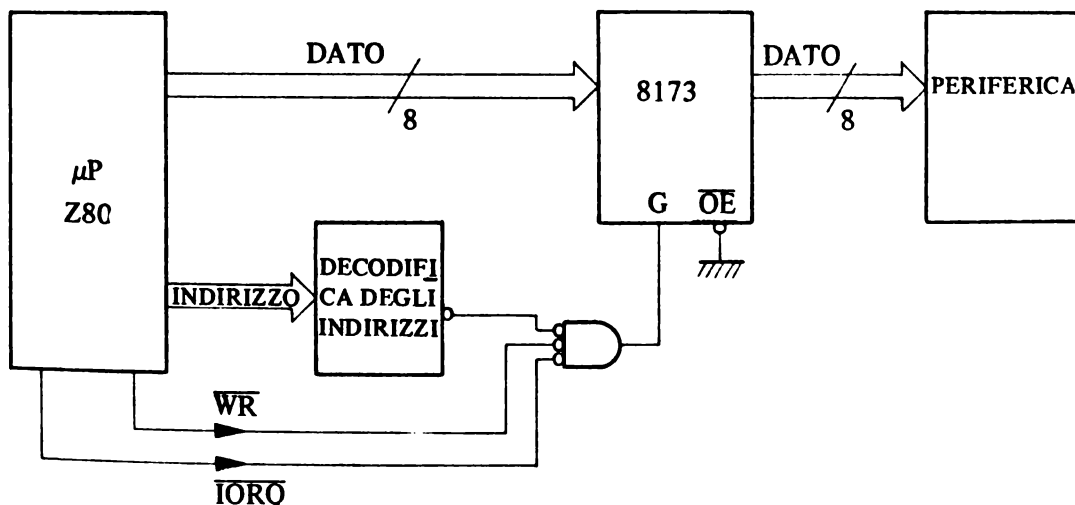



Fig. 6.8
Interfaccia di uscita realizzata con l'8173.

6.1.4. *L'8212*

In tutti i circuiti integrati presi finora in considerazione sono presenti uno o due ingressi di comando: esistono altri componenti che, sia per il maggior numero di tali ingressi, sia per qualche altra funzione che sono in grado di svolgere, permettono un più facile interfacciamento fra una periferica ed il microprocessore. Uno di questo componenti è l'8212, un buffer lacht con 8 ingressi ed 8 uscite; il suo schema logico è riportato nella fig. 6.9.

DI_1-DI_8	DATA IN
DO_1-DO_8	DATA OUT
$\overline{DS_1}$-DS_2	DEVICE SELECT
MD	MODE
STB	STROBE
$\overline{INT}$	INTERRUPT (ACTIVE LOW)
$\overline{CLR}$	CLEAR (ACTIVE LOW)

STB	MD	$(DS_1 \cdot DS_2)$	DATA OUT EQUALS
0	0	0	3 STATE
1	0	0	3 STATE
0	1	0	DATA LATCH
1	1	0	DATA LATCH
0	0	1	DATA LATCH
1	0	1	DATA IN
0	1	1	DATA IN
1	1	1	DATA IN

$\overline{CLR}$ – RESETS DATA LATCH
SETS SR FLIP FLOP
(NO EFFECT ON OUTPUT BUFFER)

CLR	$(\overline{DS_1} \cdot DS_2)$	STB	*SR	INT
0	0	0	1	1
0	1	0	1	0
1	1	⌐_	0	0
1	1	0	1	0
1	0	0	1	1
1	1	⌐_	1	0

*INTERNAL SR FLIP FLOP

Fig. 6.9
Schema logico della porta ad 8 bit 8212 (Intel Component Data Catalog 1979).

Nell'8212 due ingressi di comando $DS_1/$ e DS_2, attivi a livello logico basso e alto rispettivamente, permettono di ottenere la selezione del dispositivo; l'ingresso MD permette di stabilirne il modo di funzionamento. Esiste inoltre un ingresso di Strobe, STB, ed uno di Clear, CLR/, il quale se attivato consente di predisporre l'8212 in una determinata condizione iniziale. Il modo di funzionamento si deduce dalle tabelle di verità riportate nella fig. 6.9.
Si tratta ora di stabilire con quali condizioni si possono memorizzare i dati presenti all'ingresso e renderli disponibili all'uscita: osservando lo schema dell'8212 si può dedurre che esso viene tolto dallo stato di alta impedenza o se sono attivati i due segnali di selezione $DS_1/$ e DS_2, oppure se è attivato MD a valor logico alto.
Quanto è presente in uscita dipende dal valore assunto oltre che dai segnali sopra ricordati, anche da quello di Strobe.
Le condizioni che si possono presentare sono:

- MD attivato, $DS_1/$ e DS_2 disattivati: in tal caso in uscita è presente quanto era stato precedentemente memorizzato sui FF interni al componente;
- $DS_1/$, DS_2 attivati ed MD disattivato: in questo caso l'uscita dipende dal valore assunto dal segnale di Strobe e precisamente se questo non è attivo si è ancora nelle condizioni precedenti, mentre se è attivo le uscite DO_i assumono il valore presente agli ingressi DI_i;
- se entrambi i comandi $DS_1/$, DS_2 ed MD sono tutti attivi le uscite DO_i assumono il valore presente agli ingressi DI_i qualunque sia il valore di Strobe.

L'8212 permette anche di generare una richiesta di interruzione al verificarsi di precise condizioni: in esso è infatti presente un flip-flop, di tipo D, con l'ingresso sempre al valor logico basso e con quelli di clock e di set che dipendono da alcuni ingressi di controllo.
Il flip-flop è settato, cioè la sua uscita Q assume il valor logico alto, o quando il dispositivo è selezionato tramite $DS_1/$ e DS_2 oppure quando è attivato l'ingresso di CLEAR/; quando è settato l'uscita di richiesta di interruzioni, INT/, è portata al valore logico alto, cioè non è attiva.
La commutazione del flip-flop avviene solo se è reso attivo l'ingresso di STB: si comprende quindi come l'8212 possa essere utilizzato in modo molto semplice nel caso si debba interfacciare ad un microelaboratore una periferica che può fornire i dati solo per brevi intervalli di tempo e che comunque identifica tali intervalli con l'attivazione dello Strobe. In questo caso i dati sono memorizzati dall'8212 e contemporaneamente è richiesto il servizio al microprocessore tramite l'attivazione della richiesta di interruzione.
Una successiva lettura dell'8212, da parte della CPU per acquisire i dati in

esso memorizzati, provoca anche la commutazione del FF di interruzione e quindi l'automatica disattivazione della richiesta.
Come si è fatto notare, la presenza di più ingressi di controllo permette di utilizzare tale componente in situazioni diverse; si tratta solo di usarli in modo adeguato a seconda delle specifiche richieste.
Si può ad esempio utilizzare tale componente semplicemente come buffer unidirezionale di tipo three-state collegando gli ingressi di comando come in fig. 6.10.

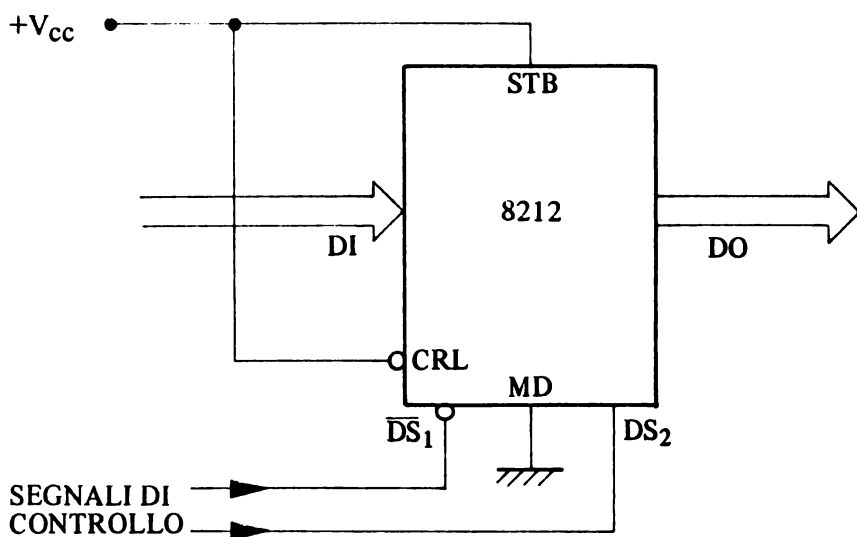



Fig. 6.10
L'8212 utilizzato come buffer unidirezionale.

L'8212 può essere tolto dallo stato di alta impedenza agendo sugli ingressi DS_1/ e DS_2: quando questi sono attivati quanto è presente all'ingresso è reso disponibile all'uscita. Si ottiene così un funzionamento simile a quello visto per l'AmZ8144, di fig. 6.2, e cioè ora l'8212 si comporta come un buffer la cui funzione potrebbe essere esclusivamente quella di poter comandare dei carichi elevati.
Dalle caratteristiche fornite dal costruttore si può infatti ricavare che mentre la corrente richiesta all'ingresso assume un valore massimo di 0,25 mA, in uscita è disponibile invece una corrente di 15mA.
L'8212 può essere utilizzato anche come interfaccia di collegamento ad una periferica di ingresso secondo lo schema riportato in fig. 6.11.

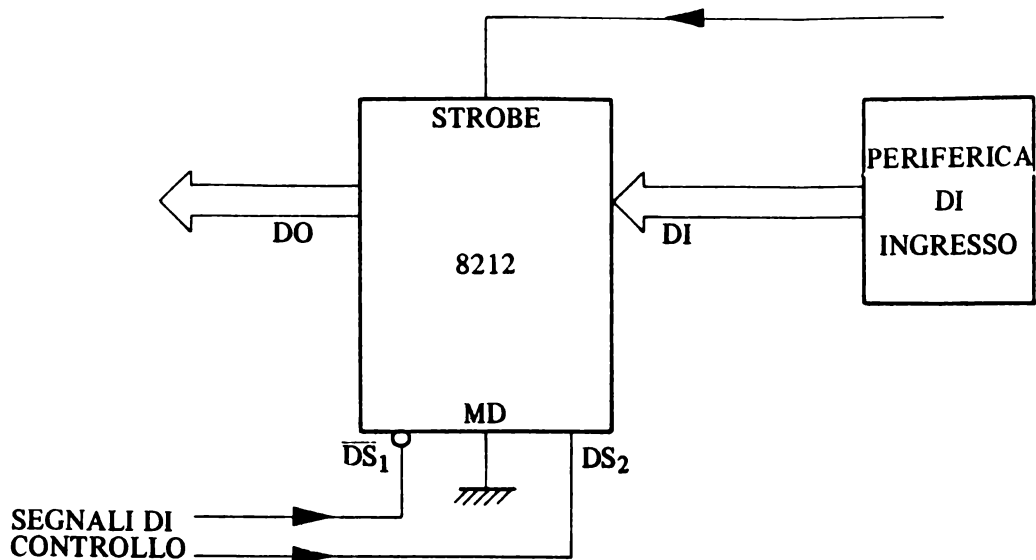



Fig. 6.11
L'8212 utilizzato come interfaccia di ingresso.

L'8212 si porta nello stato di alta impedenza se non sono attivati i due ingressi di selezione DS_1 / e DS_2; è inoltre possibile la memorizzazione di quanto inviato dalla periferica se si agisce sul segnale di Strobe, il quale può essere attivato dalla stessa periferica. L'attivazione degli ingressi di selezione fa immettere nel bus dei dati quanto è stato precedentemente memorizzato oppure quanto è presente all'ingresso: ciò a seconda del valore assunto da Strobe. Il modo con cui si attivano i vari ingressi di comando dipende ovviamente dalla particolare applicazione che si vuole realizzare.
Uno schema analogo si può utilizzare per l'interfaccia con una periferica di uscita: in questo caso l'ingresso MD è posto a livello logico alto per togliere le uscite DO_i dallo stato di alta impedenza.
Se si usano due 8212 in parallelo si può ottenere un buffer bidirezionale, il cui schema è quello riportato nella fig. 6.12.
Con l'ingresso di Strobe a livello logico alto e quello di MD basso la porta n. 2 si trova o in stato di alta impedenza o quanto è presente all'ingresso è disponibile sul bus dei dati del microelaboratore. Per rendere attivo uno solo dei due 8212 lasciando l'altro nello stato di alta impedenza si utilizza un solo ingresso di selezione, mentre l'altro è collegato ad un livello logico fisso. Quando il primo degli ingressi è a livello alto è selezionato l'8212 numero 2 e ciò corrisponde ad una operazione di lettura; per una operazione di scrittura tale ingresso deve invece essere portato a livello basso

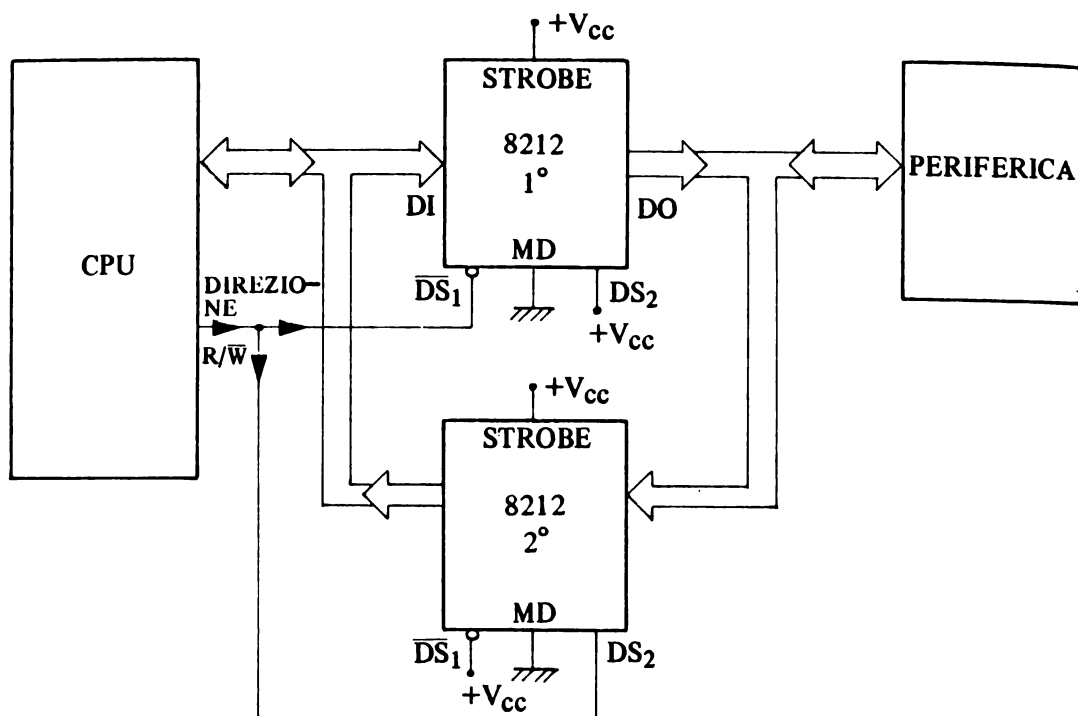



Fig. 6.12
Buffer bidirezionale realizzato con 8212.

attivando così solo l'8212 numero 1. Il segnale di selezione può essere ricavato dai segnali di controllo inviati dall'unità centrale eventualmente ricorrendo all'utilizzazione di una rete combinatoria, realizzata in modo diverso a seconda dei segnali di controllo a disposizione.
Per mantenere a disposizione i dati inviati dalla CPU alla periferica di uscita si può adottare lo schema di fig. 6.13.

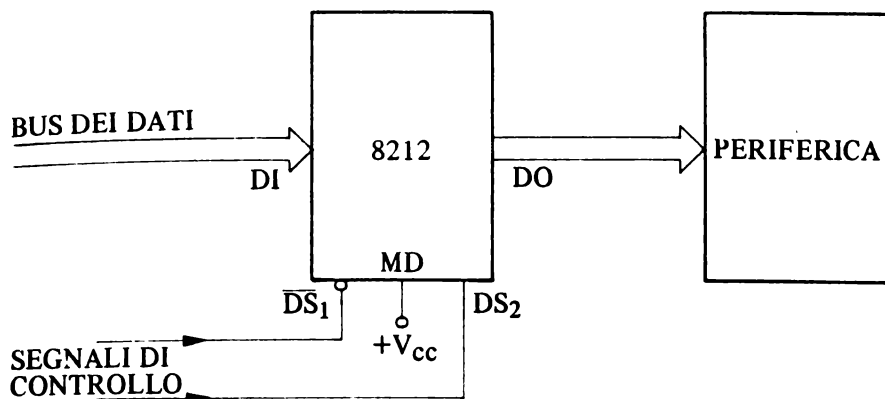



Fig. 6.13
L'8212 utilizzato come interfaccia di uscita.

Rispetto ai casi precedenti in questo schema il comando MD è posto ad un valore logico alto; quando la CPU seleziona la porta attivando i segnali $DS_1/$ e DS_2 quanto è presente sul bus dei dati è reso disponibile in uscita e contemporaneamente memorizzato sui latch interni all'8212.
Una volta disattivati i segnali $DS_1/$ e DS_2 all'uscita rimane ancora disponibile quanto è stato memorizzato, in quanto il segnale MD è a livello logico alto.
La presenza del circuito per la generazione del segnale di richiesta di interruzione può essere utile in diverse applicazioni. Un possibile schema di utilizzazione è quello riportato nella fig. 6.14.
L'attivazione del segnale di Strobe da parte della periferica attiva il segnale INT/. La CPU, quando passa a servire tale interruzione nella fase di accettazione, può richiedere o un vettore oppure una istruzione di Restart a seconda del modo in cui funziona: tale informazione può essere ottenuta dall'8212 utilizzando il segnale di accettazione. Quanto è presente all'ingresso della porta può essere prefissato collegando i diversi ingressi DI_i a livelli logici alti o bassi a seconda dell'informazione che si desidera inviare alla CPU. Si noti che la disposizione riportata nello schema può servire solamente a gestire una sola sorgente di interruzione. La presenza di più sorgenti richiederebbe una gestione delle priorità non prevista con questo componente e che quindi deve essere realizzata con una adatta struttura hardware esterna.

Si consideri ora lo schema di fig. 6.15.

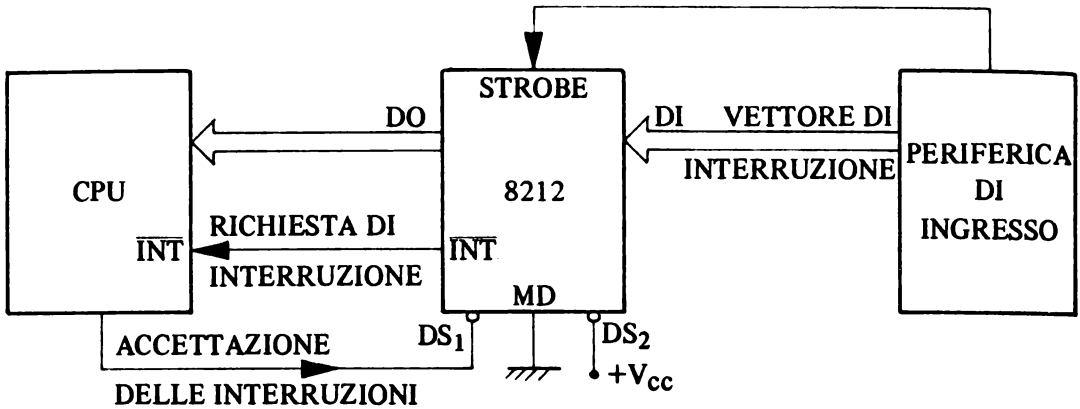



Fig. 6.14
L'8212 utilizzato come interfaccia di ingresso per il vettore di interruzione.

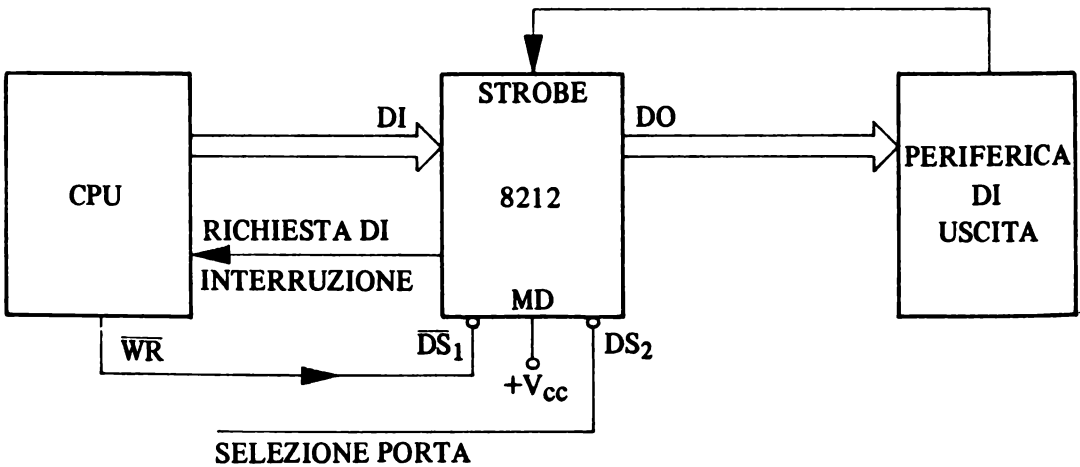



Fig. 6.15
L'8212 utilizzato come interfaccia di uscita con richiesta di interruzione.

Essendo MD posto al livello logico alto, quando la CPU invia un dato, esso è reso disponibile in uscita in forma stabile e la periferica può quindi prelevarlo quando vuole. Una volta acquisito un dato essa invia un segnale di Strobe che attiva la richiesta di interruzione: in questo caso tale segnale può anche essere interpretato come conferma da parte della periferica che il dato precedentemente inviato è stato acquisito e quindi la CPU può inviarne un altro. Con le indicazioni utilizzate nel capitolo 5 tale segnalazione corrisponde a "periferica pronta", che può anche essere interpretata come "dato ricevuto".
Il trasferimento dei dati in parallelo in un microelaboratore si presenta con notevole frequenza. I dispositivi fin qui esaminati consentono di risolvere di volta in volta i vari casi che si presentano senza tuttavia consentire di riunire in un unico componente le diverse esigenze, ciò che in qualche caso sarebbe molto utile.
Anche il più flessibile dei componenti finora esaminati, l'8212, presenta delle notevoli limitazioni: non è bidirezionale; difficilmente si riesce a gestire un sistema di priorità delle interruzioni; in pratica non sono previsti segnali di handshaking; ecc, anche se a queste esigenze si può parzialmente dare una soluzione con un hardware opportuno.
I costruttori di microprocessori hanno messo a punto dei circuiti integrati che permettono di risolvere la quasi totalità dei problemi che si presentano nella comunicazione in parallelo con periferiche di ingresso o di uscita.
In genere tali componenti sono stati previsti per essere utilizzati con un particolare microprocessore: essi infatti utilizzano direttamente i segnali di controllo forniti da quel particolare microprocessore ed il loro funzionamento è compatibile con le caratteristiche temporali di tali segnali.
Il nome di tali componenti varia da costruttore a costruttore (PIO: parallel I/O, PIA: peripheral interface adapter, PPI: programmabile peripheral interface, ecc.), ma una caratteristica fondamentale che essi hanno in comune e che li distinguono nettamente rispetto ai dispositivi finora visti è la loro programmabilità: con questo si deve intendere che il loro modo di funzionamento non è rigidamente stabilito dal costruttore, come si è visto per l'8212, ma è possibile, mediante opportune informazioni inviate dalla CPU e gestite da programma, stabilire le modalità di funzionamento.
E' altresì possibile cambiare queste modalità durante l'esecuzione del programma per adattare il microelaboratore a mutate condizioni che si potrebbero verificare.
La caratteristica della programmabilità è molto utile, dato che consente di rendere sufficentemente flessibile e adatta a multeplici scopi una struttura hardware prefissata che di per sè non si presta a modifiche.
Si analizzerà ora uno di questi componenti, previsto per essere utilizzato con il microprocessore Z80, e nel descrivere le sue caratteristiche si cercherà di mettere in risalto quelle che hanno una certa generalità rispetto a

quelle che sono imposte dall'unità centrale cui è stato previsto di collegarlo.

6.2. Lo Z80 - PIO

Lo Z80-PIO (Parallel I/O) è un dispositivo programmabile che presenta due porte di ingresso o di uscita, da 8 bit ciascuna, e che realizza una interfaccia compatibile con il microprocessore Z80.
La CPU può imporre il modo di funzionamento della PIO rendendo così possibile e semplice l'interfacciamento con periferiche diverse senza richiedere dell'hardware aggiuntivo da adattare di volta in volta a seconda del tipo di applicazione.
Lo schema a blocchi della PIO è riportato nella fig. 6.16.

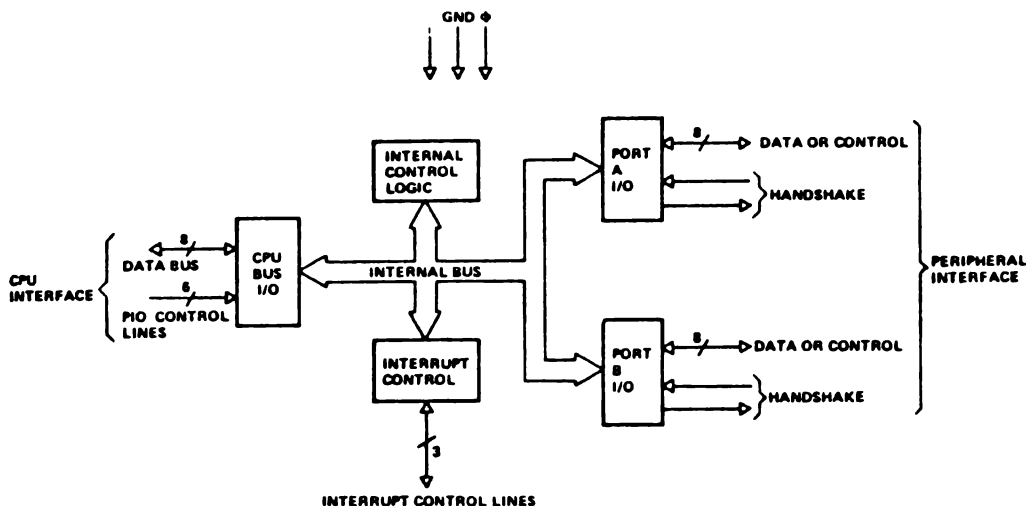



Fig. 6.16
Schema a blocchi della PIO (MOSTEK Z80 Microcomputer Data Book 1981).

In essa si può notare la presenza di 5 blocchi funzionali distinti: un primo blocco realizza l'interfaccia fra i bus dei dati e di controllo del sistema ed il componente stesso.
Il blocco di controllo gestisce, interpreta e coordina tutte le informazioni sia provenienti dalla CPU, tramite il blocco di interfaccia, sia dagli altri blocchi interni. Si noti che è il blocco di controllo, presente all'interno di ogni dispositivo programmabile, che fornisce a questi dispositivi la necessaria autonomia di funzioanmento per sgravare la CPU dai compiti di coor-

dinamento e di controllo del funzionamento delle porte di ingresso e di uscita.
I due blocchi relativi alle due porte di ingresso e/o di uscita contengono tutta la circuiteria necessaria alla gestione e dei dati provenienti dalle, o inviati alle, periferiche esterne, e dei segnali che permettono di facilitare il colloquio con le periferiche nel trasferimento delle informazioni.
E' presente infine un blocco che gestisce le richieste di interruzione e permette di risolvere le priorità nel caso siano presenti più sorgenti di interruzione.
Lo schema analizzato potrebbe essere valido per qualsiasi tipo di componente programmabile per la comunicazione in parallelo: le differenze che si riscontrano riguardano principalmente il numero di porte presenti, le modalità di programmazione, i segnali di controllo utilizzati, il modo con cui si attua la gestione delle interruzioni, ecc.
Si è già analizzato con un certo dettaglio il blocco di gestione delle interruzioni che è presente non solo nella PIO ma anche in tutti gli altri componenti della famiglia dello Z80.
Può essere interessante analizzare ora lo schema a blocchi delle due porte A e B; le differenze tra esse sono molto piccole e verranno messe in risalto man mano che si procede nell'illustrazione di questo componente: lo schema, riportato nella fig. 6.17, può essere considerato valido per entrambe le porte.

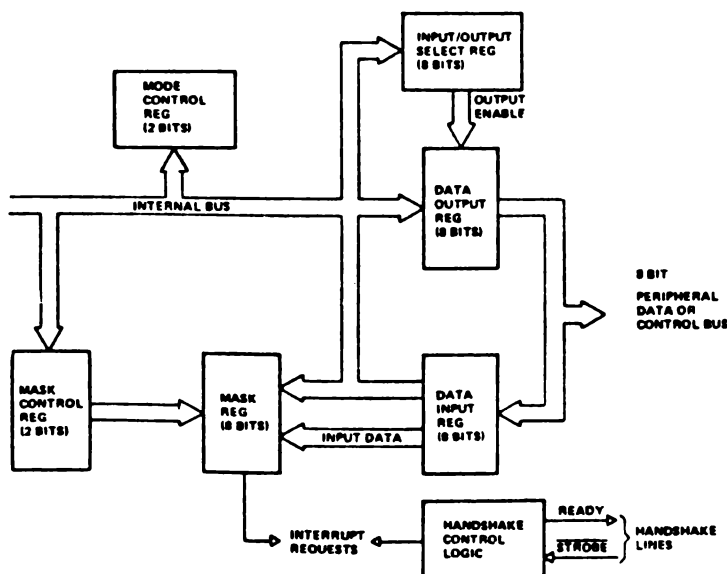



Fig. 6.17
Schema a blocchi delle porte A o B della PIO. (Z80 MOSTEK. Data Book 1981).

Come si può notare esistono un registro per i dati di ingresso ed uno per quelli di uscita; ogni porta cioè può funzionare o come porta di ingresso o come porta di uscita. La sola porta A è in grado inoltre di funzionare in modo bidirezionale.
E' anche possibile un quarto modo di funzionamento nel quale ogni singolo bit di ciascuna delle due porte può essere programmato in modo indipendente per funzionare o come ingresso o come uscita.
I quattro diversi modi di funzionamento possono essere programmati variando il contenuto, di due bit del "mode control register". I registri "Input/Output Select", "Mask Control" e "Mask" sono utilizzati nel quarto modo di funzionamento, come sarà illustrato in seguito. Esiste inoltre un ultimo blocco che comunica con l'esterno mediante due segnali "Ready" e "Strobe", di handshaking, equivalenti a quelli già visti nel capitolo 5 dove erano stati indicati col nome di "dato pronto" o "dato ricevuto" oppure di "periferica pronta".

6.3. I segnali della PIO

Nella fig. 6.18 è riportata la configurazione dei piedini della PIO: essi sono raggruppati in base alle caratteristiche funzionali.
I piedini a sinistra nella figura, ed i segnali ad essi associati, servono per mettere in comunicazione la PIO con la CPU, mentre quelli a destra sono destinati al collegamento con le unità esterne.
Tra i primi si possono individuare gli otto piedini D0-D7, da collegare al bus dei dati, e quelli associati al controllo che svolgono le funzioni di seguito descritte:

PORT B/A SEL: agendo su questo ingresso la CPU seleziona quale delle due porte è interessata al trasferimento di dati; in particolare quando è posto a livello logico alto è selezionata la porta B, in caso contrario la A.

CONTROL/DATA SEL: ponendo questo ingresso a livello alto la CPU informa la PIO che il dato presente sul data bus è destinato all'unità di controllo, interna alla PIO stessa, mentre se tale segnale è a livello logico basso significa che sul bus di dati è presente una informazione che deve essere interpretata come dato.

CHIP ENABLE/: un segnale a valore logico basso su questo piedino abilita la PIO ad accettare comandi o dati dalla CPU durante un ciclo di scrittura o di trasmettere dati alla CPU durante un ciclo di lettura.

Esistono poi tre ingressi, M1/, IORQ/ e RD/, che sono gestiti dallo Z80, e servono per individuare con le loro combinazioni le operazioni di lettura o di scrittura, oppure di accettazione di una richiesta di interruzione o di reset della PIO stessa.

M1/: il segnale presente a questo ingresso, attivo a livello logico basso, è generato dalla CPU come segnale di sincronismo per il controllo delle di-

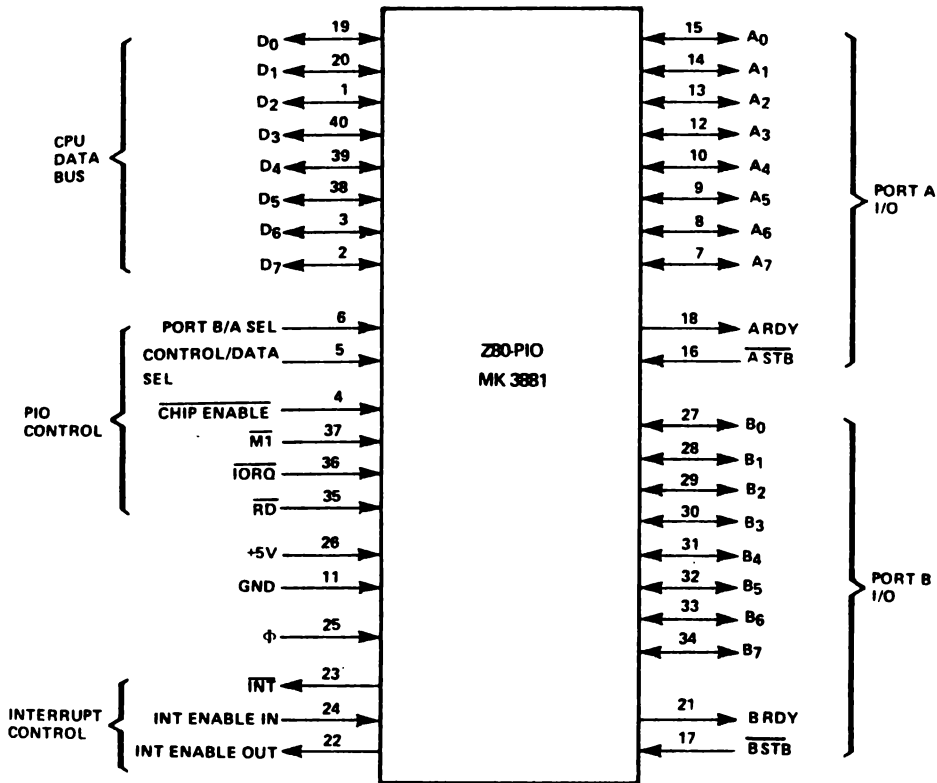



Fig. 6.18
I segnali della PIO (MOSTEK Z80 Microcomputer Data Book 1981.

verse operazioni interne alla PIO. Quando, oltre a questo, è attivato anche l'ingresso RD/ ciò significa che è in esecuzione un ciclo di fetch di un codice operativo di una istruzione.
Se invece risultano contemporaneamente attivi M1/ e IORQ/, ciò significa che la CPU ha accettato una richiesta di interruzione e si aspetta dal dispositivo, che ha generato la richiesta, il codice operativo di una istruzione di RESTART, oppure, se si è predisposto il modo di funzionamento 2 per le interruzioni, il vettore che permette di individuare nella tabella delle interruzioni l'indirizzo di partenza della relativa routine di servizio.
Per la PIO il segnale M1/ ha anche altre due funzioni: oltre a fornire un segnale di sincronismo alla logica di gestione delle interruzioni, esso permette di far entrare la PIO nello stato di Reset e quindi di stabilire una determinata condizione di partenza. Affinché ciò accada è necessario che risulti attivo solamente il segnale M1/, mentre IORQ/ e RD/ devono trovarsi a livello logico alto.

IORQ/: questo segnale di ingresso è attivo a livello logico basso ed è usato assieme ai segnali B/A SEL, C/D SEL/, CE/, ed RD/ per trasferire comandi o dati tra la PIO e la CPU Z80. Si possono presentare diverse situazioni. Se CE/ RD/ e IORQ/ sono tutti attivi e l'ingresso C/D SEL si trova a livello logico basso, la porta individuata dal valore del segnale B/A SEL trasferirà dati alla CPU. Se invece tutti gli altri segnali rimangono allo stesso valore mentre RD/ è a livello logico alto allora la porta, individuata sempre dal segnale B/A SEL, è interessata ad una operazione di scrittura di dati da parte della CPU. Nel caso la CPU voglia invece inviare un comando durante la programmazione della PIO, essa deve portare a livello logico alto il segnale C/D SEL.

RD/: questo segnale di ingresso, generato dalla CPU, indica che essa sta seguendo un ciclo di lettura, quando è a livello logico basso. Se invece è a livello logico alto l'operazione in corso è una scrittura; è ovvio che affinché l'operazione riguardi la PIO è necessaria la contemporanea attivazione di IORQ/, e di CE/.

E' inoltre presente anche un segnale ϕ, di clock, generato anch'esso dalla CPU, il quale permette di sincronizzare il funzionamento della PIO con la CPU stessa.
Esistono infine tre segnali, che permettono la gestione delle interruzioni, il cui significato è stato descritto nel capitolo relativo alle interruzioni.
Per quanto riguarda invece i segnali che servono al collegamento con delle periferiche, ognuna delle due porte ne presenta otto, associati ad altrettanti piedini, che possono essere o di ingresso o di uscita, a seconda di come è stata predisposta a funzionare quella porta.
I segnali di handshaking STB/ e RDY assumono significati diversi a seconda del modo di funzionamento della porta interessata.
Per il segnale STB/, i vari significati che esso può assumere possono essere così riassunti:

a) se il modo di funzionamento è di ingresso, questo segnale è inviato dalla periferica per memorizzare un dato nel registro di ingresso della PIO, in modo analogo a quello visto per l'8212.

b) se il modo selezionato è invece di uscita il fronte di salita di questo segnale, inviato sempre dalla periferica, indica che è stato letto il dato presente sul registro di uscita della porta interessata.

c) nel modo di funzionamento bidirezionale sulle linee di uscita della porta è disponibile quanto memorizzato nel registro di uscita per tutto il tempo in cui questo segnale è attivo. Anche in questo caso il fronte di salita ha il significato che la periferica ha prelevato il dato;

d) nel funzionamento in control mode questo segnale non ha significato.

Anche per il segnale RDY/, di uscita ed attivo a livello logico alto, il si-

gnificato varia a seconda del modo di funzionamento selezionato:

a) se il modo è di ingresso questo segnale è attivo quando il registro di ingresso della porta è vuoto e pronto ad accettare un dato dalla periferica;
b) se il modo selezionato è di uscita questo segnale diventa attivo per indicare che il registro di uscita contiene un dato e che le linee di uscita della porta sono stabili e pronte per il trasferimento del dato alla periferica;
c) nel modo di funzionamento bidirezionale questo segnale è attivo quando un dato è disponibile sul registro di uscita per il trasferimento alla periferica; si noti che il dato non è posto sulle linee di uscita a meno che il segnale di STROBE/ non sia attivo.
d) nel funzionamento in control mode questo segnale è forzato dalla PIO allo stato logico basso.

6.4. Modi di funzionamento della PIO

Per programmare la PIO è necessario siano inviati, da parte della CPU e tramite il bus dei dati, dei byte di comando, il cui contenuto permette di

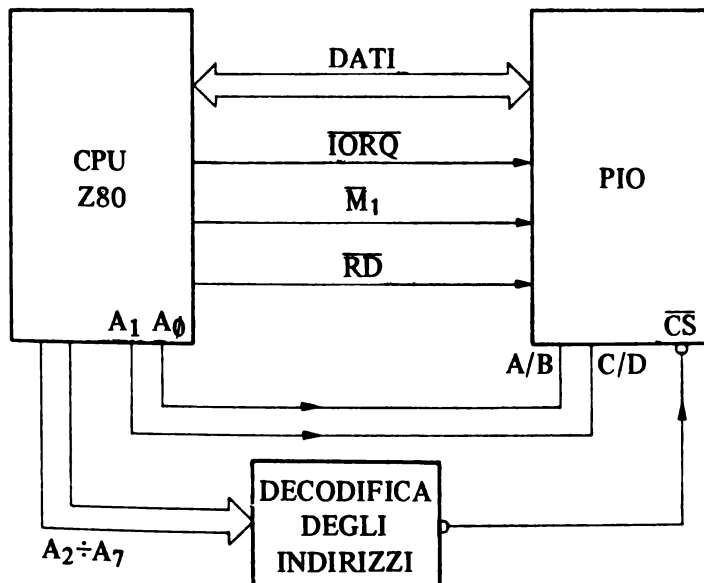



Fig. 6.19
Collegamento della PIO allo Z80.

definire le diverse particolarità del modo di funzionamento che si desidera ottenere. La CPU deve quindi attivare la PIO, e precisare sia a quale delle due porta essa si riferisce, sia se quanto contenuto nel bus dei dati deve essere interpretato come un comando o come un dato. Tutte queste informazioni sono fornite attivando in modo opportuno i segnali di controllo prima esaminati e precisamente B/A SEL e C/D SEL, oltre naturalmente al segnale CS/ necessario per selezionare la PIO stessa. Si può ad esempio adottare lo schema della fig. 6.19.
Come si vede il bit A_0 degli indirizzi è collegato all'ingresso di selezione delle due porte, mentre A_1 a quello che serve per distinguere fra comandi e dati. Infine il segnale di CS/ è attivato da una opportuna decodifica dei restanti 6 bit più significativi di indirizzo. In questo modo la identificazione delle diverse operazioni di I/O avviene come riportato nella tabella 6.1:

Tabella 6.1

A_1 = C/D SEL	A_0 = B/A SEL	
0	0	Dato relativo alla porta A
0	1	Dato relativo alla porta B
1	0	Comando per la porta A
1	1	Comando per la porta B

Gli altri 6 bit di indirizzo saranno utilizzati o meno a seconda dal tipo di decodifica adottato. Si noti che da quanto indicato nello schema il componente è selezionato con quattro indirizzi diversi ognuno dei quali serve per un ben preciso motivo. Ad esempio se la CPU esegue l'istruzione:

OUT XXXX XX10 (indirizzo)

essa invierà alla porta A, attraverso il bus dei dati, un byte che deve essere interpretato come un comando; analogamente per gli altri casi.
Per programmare il funzionamento di ciascuna delle due porte è necessario inviare alla porta interessata dei byte di comando, la cui configurazione è stabilita dal costruttore e deve essere seguita rigidamente.
Si deve innanzi tutto stabilire il modo di funzionamento che si desidera per quella determinata porta: il byte di comando che si deve inviare assume il seguente formato: (fig. 6.20).

D_7			D_4	D_3			D_0
M_1	M_0	X	X	1	1	1	1

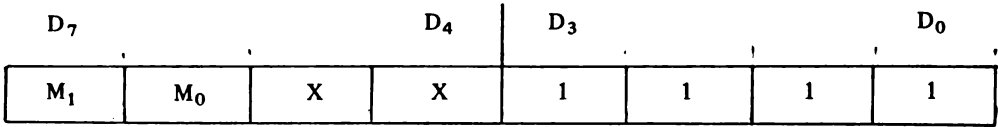



Fig. 6.20
Struttura del byte di comando del modo per la PIO.

I quattro bit meno significativi devono essere sempre a 1, mentre i bit D_4 e D_5 non hanno significato in questo caso e possono assumere qualsiasi valore. Gli ultimi due bit D_6 e D_7 sono proprio quelli che permettono di individuare uno dei quattro modi di funzionamento possibili secondo la tabella 6.2:

Tabella 6.2

D_6	D_7	
0	0	modo 0 (uscita)
0	1	modo 1 (ingresso)
1	0	modo 2 (bidirezionale)
1	1	modo 3 (controllo)

6.4.1. *Modo 0*

Nel modo di funzionamento 0, o di uscita, un qualsiasi dato inviato dalla CPU ad una delle porte della PIO è da essa memorizzato sul registro di uscita associato a quella porta (fig. 6.17) e reso immediatamente disponibile sulle corrispondenti linee. Il contenuto di tale registro può essere cambiato in qualsiasi momento dalla CPU semplicemente scrivendo un nuovo dato su quella porta. E' da notare che il contenuto del registro di uscita può anche essere letto dalla CPU con una normale operazione di lettura di quella porta. L'invio di un dato ad una porta funzionante nel modo 0 fa si che sia attivato anche il relativo segnale READY, per avvertire la periferica che un nuovo dato è disponibile, facilitando quindi il trasferimento di informazioni. Tale segnale rimane attivo finché la periferica non avvisa che ha prelevato il dato ricevuto mediante l'attivazione dell'ingresso STROBE/.
Supponendo che la porta sia stata abilitata a fare richieste di interruzione, l'attivazione del segnale STROBE/ provoca contemporaneamente una richiesta di interruzione alla CPU.
Come si può notare l'utilizzazione della PIO rende decisamente più semplice la struttura hardware richiesta per il microelaboratore rispetto all'utilizzazione dei buffer prima esaminati.

Il diagramma temporale dei vari segnali che interessano in questo caso è riportato nella fig. 6.21.

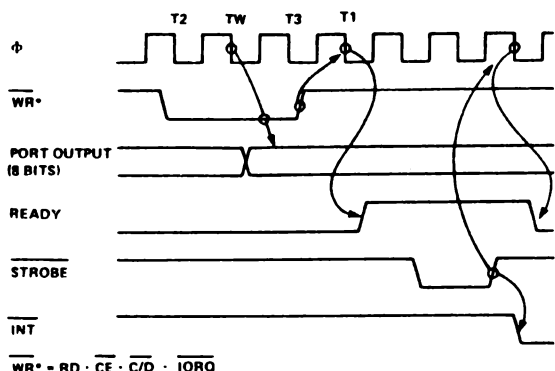
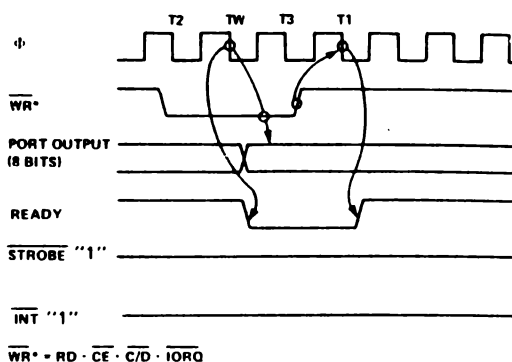



Fig. 6.21
Diagrammi temporali relativi al modo di funzionamento 0 della PIO (MOSTEK, Z80 Microcomputer Data Book 1981).

Il segnale di riferimento temporale è quello di clock, ϕ; il segnale WR/* è generato internamente alla PIO in funzione di alcuni segnali di controllo, come indicato in fig. 6.21. Come si può notare il segnale READY è attivato quando sicuramente i dati sono disponibili in uscita, mentre il fronte di salita di STROBE/, generato dalla periferica, provoca la disattivazione di READY e l'attivazione della richiesta di interruzione in sincronismo con il segnale di clock di riferimento.
Non sempre la periferica collegata è in grado di usare i segnali di handshaking per il trasferimento dei dati, oppure non è in grado di generare il segnale di STROBE/. In questo caso dopo la prima scrittura, dato che il segnale STROBE/ non è mai attivato, il segnale READY rimane sempre a livello logico alto.
All'invio di un nuovo dato da parte della CPU cambiano per un certo intervallo di tempo i livelli logici presenti alle uscite della porta: se la periferica procede alla acquisizione durante questo transitorio ovviamente essa riceve una informazione errata. La PIO prevede anche questa eventualità: se all'arrivo di un nuovo dato trova il segnale READY attivo, provvede automaticamente a disattivarlo per due periodi di clock in modo da inibire l'acquisizione durante il transitorio. Quanto detto è illustrato nel diagramma temporale di fig. 6.21b.

Come si è detto, una volta selezionato il modo di funzionamento 0, il contenuto del registro di uscita è immediatamente disponibile sulle rispettive linee di uscita; quando, all'inizio, è fornita alla PIO la tensione di alimentazione tale contenuto è indeterminato per cui lo è anche quello delle linee di uscita. Non sempre una tale situazione è accettabile: è possibile però caricare il registro di uscita della porta anche se non è stato selezionato il modo di funzionamento. Così facendo al momento della selezione del modo 0 le linee di uscita assumono lo stato determinato in precedenza.

6.4.2. *Modo 1*

Nel modo di funzionamento 1 la porta è di ingresso e quindi i dati sono inviati dalla periferica e letti dalla CPU.
I due segnali di handshaking ora assumono un significato diverso rispetto al caso precedente e possono essere utilizzati o meno, a seconda del tipo di periferica.
Si supponga di utilizzare i due segnali di handshaking e si faccia riferimento al diagramma temporale di fig. 6.22.

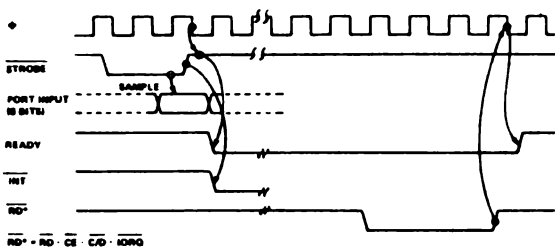
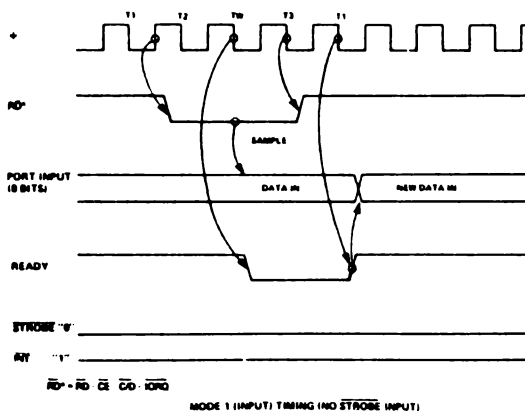
MODE 1 (INPUT) TIMING

MODE 1 (INPUT) TIMING (NO STROBE INPUT)

a) b)

Fig. 6.22
Diagrammi temporali relativi al funzionamento in modo 1 della PIO (MOSTEK Z80 Microcomputer Data Book 1981)

Si supponga che la periferica presenti il dato sui piedini associati alla porta interessata e attivi il segnale di STROBE/ che in questo caso nei confronti della CPU assume il significato di "dato pronto". Il dato è memorizzato

sul registro di ingresso della porta e sul fronte di STROBE/ è disattivato il segnale READY, che in tale modo di funzionamento assume il significato di "dato ricevuto". Inoltre è attivata la richiesta di interruzione, se la PIO era stata precedentemente a ciò abilitata.
A questo punto spetta alla CPU accettare la richiesta fatta e procedere alla lettura del nuovo dato inviato. L'operazione di lettura è indicata nel diagramma temporale mediante il segnale RD* dal significato analogo a quello visto per il segnale WR* in fig. 6.21; esso è attivo quando lo sono gli ingressi RD/, CE/, IORQ/ e C/D è a livello basso.
Il dato, memorizzato sul registro della porta, è quindi letto dalla CPU; sul fronte di salita di RD/ si ha l'attivazione del segnale READY per indicare alla periferica che il dato precedentemente inviato è stato acquisito e che la porta è disponibile a riceverne un altro.
Nel caso la periferica non sia in grado di utilizzare i segnali di handshaking, si deve organizzare opportunamente la lettura come in un normale buffer. Il relativo diagramma temporale è riportato nella fig. 6.22b.
Quando è attivato il segnale RD/, il segnale READY si trova attivo in quanto nessun dispositivo esterno ha inviato il segnale di STROBE/. Onde evitare che durante l'acquisizione da parte della CPU la periferica invii un nuovo dato, per cui si avrebbe una informazione indeterminata, è disattivato READY per 2 periodi di clock ed è permesso alla periferica di inviare un nuovo dato solamente con il segnale READY ritornato attivo.
Dopo un RESET la PIO si predispone in una ben determinata configurazione: in particolare il segnale READY è posto a livello logico basso. In questa situazione non è possibile abbia inizio il trasferimento di dati con segnali di handshaking. Infatti la periferica non invia un dato, in quanto READY è a livello basso, per cui non viene generata alcuna richiesta di interruzione e quindi la CPU non esegue una lettura.
E' necessario allora che la CPU esegua una operazione di lettura il cui solo scopo è di attivare READY, dopo di che la periferica può inviare un dato e quindi si può iniziare il trasferimento.

6.4.3. *Modo 2*

Il modo di funzionamento 2 permette un trasferimento di dati bidirezionale attraverso la sola porta A, mentre la porta B necessariamente deve essere predisposta a funzionare nel modo 3. I segnali di handshaking della porta B sono utilizzati ora come tali per la porta A, in ingresso, mentre quelli della porta A sono di handshaking per l'uscita dei dati.
Si può considerare tale modo di funzionamento come l'insieme dei due modi visti in precedenza, con lievi differenze per tener conto che dalla stessa porta possono transitare dati nelle due direzioni.
Il diagramma temporale di tale modo è riportato in fig. 6.23.

PORT A, MODE 2 (BIDIRECTIONAL) TIMING

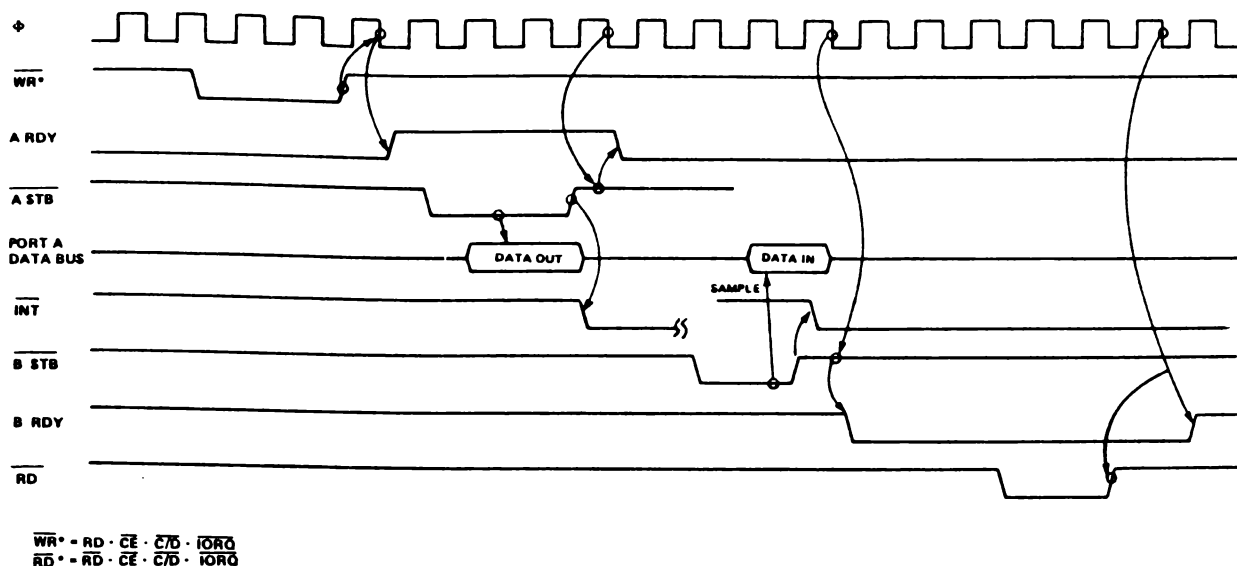



Fig. 6.23
Diagramma temporale relativo al modo di funzionamento 2 della PIO. (MOSTEK, Z80 Microcomputer Data Book 1981).

Il dato è reso disponibile in uscita solamente se è attivato il segnale STROBE/A: è la periferica quindi, che, verificato il segnale READY A, impone alla PIO di presentare il contenuto del registro di uscita sulle linee omonime, e ciò avviene solamente finchè essa mantiene attivo il segnale di STROBE/A.
Nei confronti invece dell'operazione di lettura in pratica si ha un comportamento uguale a quello visto per il modo 1.

6.4.4. *Modo 3*

Il modo di funzionamento 3, detto anche di controllo, serve in particolare in quelle situazioni in cui le informazioni che devono essere trasferite possono essere codificate con uno o pochi bit. Nel Modo 3 si può stabilire se il singolo bit, e non l'intera porta, deve essere di ingresso o di uscita: ogni bit di una porta è indipendente dagli altri. Non sono ovviamente necessari in questo caso i segnali di handshaking ed è questo il motivo per cui nel Modo 2 è necessario programmare la porta B in Modo 3, così da lasciare

liberi i due segnali di handshaking ad essa relativi.
Per programmare il modo di funzionamento 3 sono necessarie ulteriori informazioni per stabilire le condizioni di ingresso e di uscita di ogni singola linea.
E' perciò necessario, dopo che è stato inviato il byte di comando che informa una porta di predisporsi a funzionare nel modo 3, inviare un successivo byte il quale è interpretato dalla porta come indicazione di quali linee devono funzionare come ingresso e quali come uscita. Il formato di tale byte è indicato nella fig. 6.24.

D7	D6	D5	D4	D3	D2	D1	D0
I/O_7	I/O_6	IO/$_5$	I/O_4	I/O_3	I/O_2	I/O_1	I/O_0

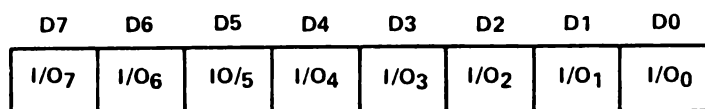



Fig. 6.24
Formato del byte di comando della direzione nel modo 3 (Mostek Z80 Microcomputer Data Book 1981)

Il valore 1 predispone la linea corrispondente in ingresso, mentre il valore 0 la predispone in uscita.
Il diagramma temporale di una operazione di lettura è riportato nella fig. 6.25.

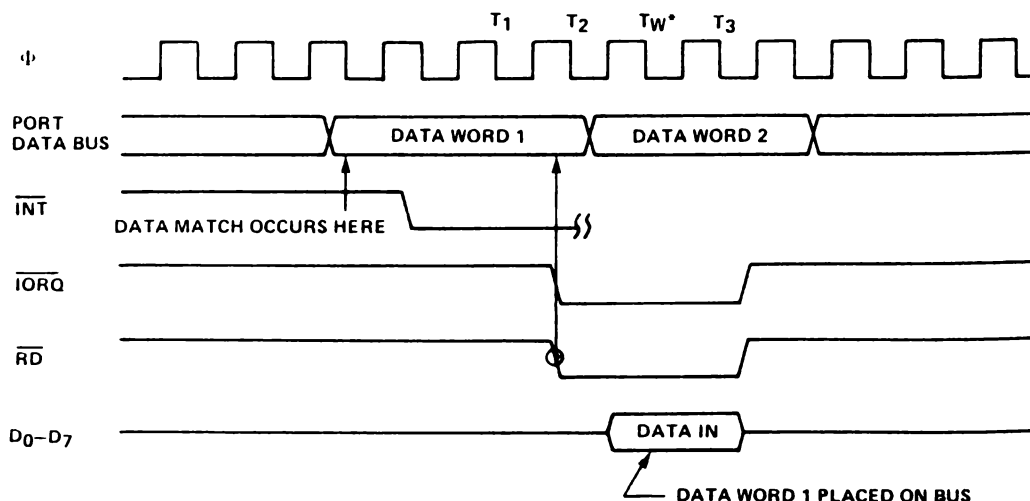



Fig. 6.25
Diagramma temporale relativo ad una operazione di lettura di una porta PIO funzionante in modo 3 (Mostek Z80 Microcomputer Data Book 1981).

I dati che sono presenti sulle linee che collegano la periferica sono acquisiti dalla CPU sul fronte di discesa del segnale di RD/; si noti che la CPU acquisisce un byte per cui il dato letto contiene le informazioni relative ai valori assunti sia dalle linee di ingresso sia da quelle di uscita.

6.5. Programmazione delle interruzioni nella PIO

Esistono varie situazioni per le quali una PIO può inviare delle richieste di interruzione; si ricorda che la PIO, come gli altri componenti periferici della famiglia dello Z80, è predisposta per la gestione di interruzioni con la CPU funzionante nel modo 2, nel quale è richiesto un vettore da cui ricavare l'indirizzo di partenza della routine di servizio.
Deve perciò essere fornito alla PIO, da programma, sia il vettore di interruzione sia informazioni sulle condizioni che devono essere soddisfatte per generare una richiesta di interruzione.
L'invio da parte della CPU, durante la programmazione della PIO, del vettore delle interruzioni avviene con una normale operazione di scrittura sulla PIO, con l'ingresso C/D portato a livello logico alto e con quello A/B al valore che identifica la porta cui il vettore si riferisce; occorre inoltre che l'informazione inviata abbia a zero il bit meno significativo. Il formato del byte da inviare diventa allora (fig. 6.26):

D7	D6	D5	D4	D3	D2	D1	D0
V7	V6	V5	V4	V3	V2	V1	0

Fig. 6.26
Formato del byte di controllo per l'invio del vettore delle interruzioni in una PIO. (Mostek Z80 Microcomputer Data Book 1981)

Poichè il bit meno significativo deve essere zero, l'indirizzo nella tabella delle interruzioni, dove è memorizzato l'indirizzo di partenza della routine di servizio, deve trovarsi in posizione pari. Questa particolarità, comune anche agli altri componente della famiglia Z80, deve essere tenuta presente in sede di programmazione. La parola di controllo, che fornisce le ulteriori necessarie informazioni sulle condizioni con cui la PIO effettua una richiesta di una interruzione è riportata nella fig. 6.27.

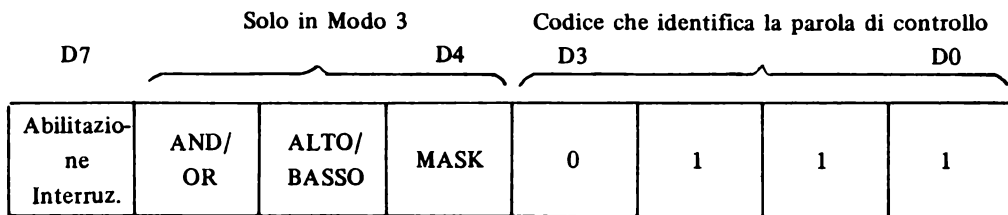



Fig. 6.27
Formato del byte per la specifica delle condizioni di interruzione.

I quattro bit meno significativi debbono assumere una particolare configurazione stabilita dal costruttore mediante la quale la PIO può distinguere quale delle diverse parole di controllo è stata inviata.
Se si pone al valore logico 1 il bit più significativo, la PIO è abilitata a gegenerare delle richieste di interruzione. Le condizioni per cui si effettuano tali richieste, nel modo di funzionamento 0, 1, 2, sono già state illustrate nell'analisi dei relativi diagrammi temporali: si tratta sempre dell'attivarsi di uno dei segnali di handshaking. Nel caso invece la porta funzioni in modo 3 è possibile specificare le varie condizioni con i bit D6, D5, D4 della parola di controllo di fig. 6.27.
Si può stabilire se le linee di ingresso devono essere considerate attive a livello logico alto oppure basso: questa opzione è indicata alla PIO mediante il valore del bit D5; se esso ha il valore 1 si considerano i segnali di ingresso attivi a livello alto, se 0 a livello basso.
Si può inoltre stabilire il tipo di funzione logica che la PIO deve calcolare sui segnali di ingresso per attivare una richiesta di interruzione; ciò è ottenuto mediante il bit D6: se questo assume il valore 1 l'operazione da eseguire è un AND se 0 un OR. Si deve ricordare che la richiesta è attivata solamente quando da una condizione falsa si passa ad una vera.
Si supponga ad esempio di avere programmato la PIO a richiedere interruzione per i segnali a livello alto e per l'operazione OR: se inizialmente tutte le linee sono a zero e da un certo istante una linea si porta a 1, si ha una richiesta di interruzione. Se poi anche un'altra linea si porta a livello alto non è generata una ulteriore richiesta in quanto questo non comporta un passaggio da falso a vero della funzione OR. Per avere una nuova richiesta di interruzione è necessario che tutte le linee si riportino a livello basso.
In fig. 6.28 è riportato l'andamento temporale dei segnali che interessano supponendo che concorrano all'attivazione delle interruzioni solamente i segnali D0 e D1.

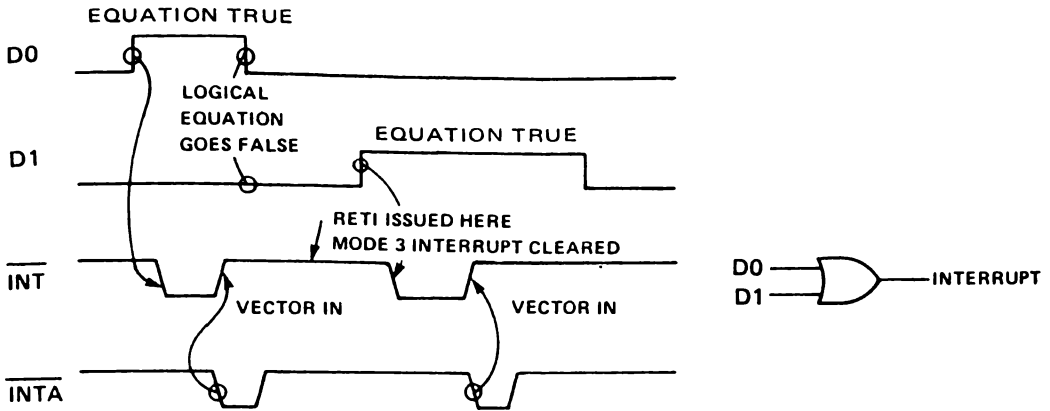



Fig. 6.28
Attivazione di una richiesta di interruzione mediante la PIO funzionante in modo 3 (Mostek, Z80 Microcomputer Data Book 1981)

Per una corretta gestione delle interruzioni devono essere soddisfatte due condizioni: la prima consiste nel fatto che D1 deve diventare attivo dopo che è andato a livello basso D0 in modo da permettere che la funzione logica diventi falsa.
E' inoltre necessario che D1 si trovi a livello alto anche dopo che è stata eseguita la istruzione RETI relativa alla routine di servizio per la richiesta fatta da D0.
Non sempre è conveniente che tutte le linee concorrino per la generazione di una richiesta di interruzione. Si possono mascherare quei bit che non devono far sentire la loro influenza: per ottenere ciò si deve porre ad 1 bit D4 di fig. 6.27; in tal modo si indica che il successivo byte di comando deve essere interpretato da parte della PIO come il byte per la mascheratura dei vari bit che non devono concorrere alla interruzione. Il formato di tale byte risulta (fig. 6.29):

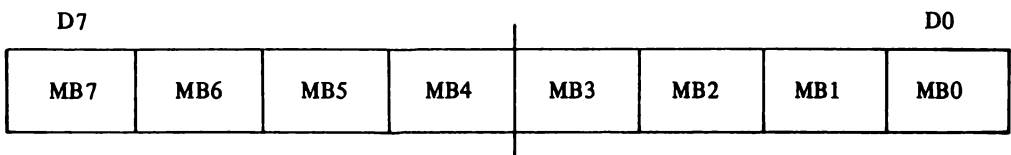



Fig. 6.29
Formato del byte di maschera per le interruzioni.

Solamente quelle linee il cui bit nella maschera assume il valore 0 partecipano alla richiesta delle interruzioni.
E' da aggiungere che il flip-flop, interno alla PIO, usato per la abilitazione o meno delle richieste di interruzione può essere settato o resettato, senza modificare il comportamento della PIO nelle altre sue funzioni, con l'ulteriore parola di controllo indicata in fig. 6.30.

D7			D4	D3			D0
Abilit.	X	X	X	0	0	1	1

Fig. 6.30
Formato del byte di controllo per l'abilitazione delle richieste di interruzione.

Come si può notare il bit più significativo permette di abilitare o meno le richieste di interruzione: se vale 1 esse sono abilitate, se vale 0 disabilitate. I primi quattro bit meno significativi devono assumere la configurazione indicata mentre gli altri tre non sono presi in considerazione dalla PIO.
Si deve porre particolare attenzione quando si invia il comando per disabilitare le richieste di interruzione: si può verificare infatti che proprio mentre la CPU sta inviando questo comando, ad esempio mediante una istruzione di OUT, la PIO generi una richiesta di interruzione. Alla fine dell'istruzione di OUT la CPU, trovando attivo il suo ingresso INT/, genererà il segnale di accettazione della richiesta e si aspetta di ricevere il vettore per la determinazione dell'indirizzo della routine di servizio. Tale vettore però non può più essere inviato dalla PIO, in quanto essa è stata appena disabilitata e quindi non ha preso in considerazione la segnalazione di accettazione della richiesta. E' chiaro allora che la CPU interpreterà come vettore quanto è presente sul data bus al momento dell'acquisizione, informazione indeterminata che ovviamente conduce ad errori. Una soluzione a questo problema consiste nel disabilitare la CPU ad accettare interruzioni mediante una istruzione DI inviare quindi il comando alla PIO per la sua disabilitazione e successivamente riabilitare la CPU ad accettare interruzioni. La sequenza delle istruzioni potrebbe ad esempio essere:

LD	A,03H;	la parola di controllo 0000 0011 è preparata nell'accumulatore
DI	;	è disabilitata la gestione delle interruzioni
OUT	(PIO), A ;	è inviata alla PIO la parola di controllo 0000 0011 per disabilitare a far richiesta di interruzioni
EI	;	è riabilitata la gestione delle interruzioni da parte della CPU.

6.6. Programmi di inizializzazione per la PIO

Si presentano alcuni semplici programmi per la inizializzazione della PIO a funzionare in uno dei modi possibili.
Si supponga di adottare lo schema di fig. 6.31:

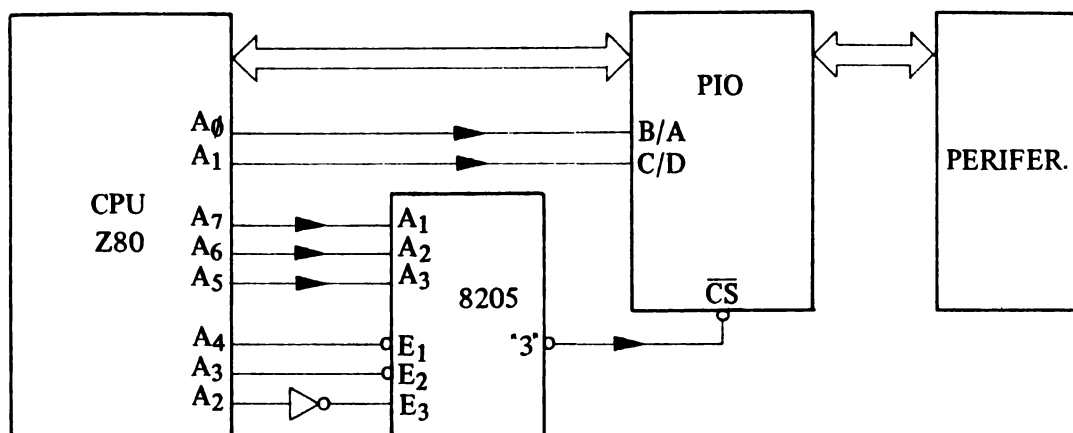



Fig. 6.31
Circuito di selezione per la PIO con indirizzi C0 - C3.

per cui gli indirizzi relativi alla PIO risultano:

C0H	porta A dato
C1H	porta A comando
C2H	porta B dato
C3H	porta B comando

Si voglia usare la porta B della PIO come buffer di uscita: la sequenza di istruzioni che possono essere usate sono:

LD	A,0FH	predispone il comando 00001111 per il modo ϕ di funzionamento
OUT	(C3H), A	invia il comando alla porta B
LD	A,. . .	caricamento in A del dato da inviare in uscita.
OUT	(C2H),A	invio del dato alla porta B.

Si voglia ora utilizzare la stessa porta impiegando anche i segnali di handshaking e la richiesta di interruzioni. Si supponga che la tabella degli indirizzi per le interruzioni abbia indirizzo iniziale TAB già predisposto in posizione pari; inoltre la routine di servizio inizi da ROUTS. Si deve innanzitutto predisporre la gestione delle interruzioni:

```
IM2                      lo Z80 risponde alle interruzioni nel modo 2

LD     HL, TAB           si carica nel registro interno I il byte più
LD     A, H              significativo della tabella TAB degli indi-
LD     I, A              rizzi per le interruzioni

LD     IX, ROUTS         si carica in TAB+4 e TAB+5 l'indirizzo di
LD     (TAB+φ4), IX      partenza della routine di servizio ROUTS

LD     A, φ4H            carica in B il vettore delle interruzioni
                         (pari)

OUT    (C3H), A          lo invia alla PIO, porta B
```

si sono così predisposte la PIO e la CPU con le informazioni necessarie per la corretta gestione delle interruzioni.
Si tratta ora di fissare il modo di funzionamento della PIO. Nell'ipotesi di voler ancora la porta B in uscita si utilizzano le istruzioni:

```
LD     A,OFH             predispone modo φ
OUT    (C3H), A          porta B in modo φ
LD     A,87H             abilitazione delle interruzioni (1000 0111)
OUT    (C3H),A
```

Se invece si vuole utilizzare la porta B per l'ingresso dei dati basta semplicemente sostituire il comando di predisposizione del modo di funzionamento;

```
LD     A, 4FH            modo 1: 01000 1111
OUT    (C3H), A
```

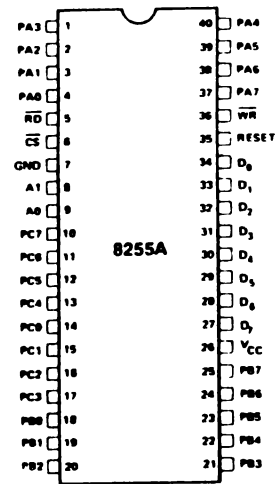
e analogamente nel modo 2 nel qual caso si devono però predisporre due routine di servizio. Nel modo 3 le informazioni da inviare sono in numero maggiore. Si supponga di volere i quattro bit più significativi di uscita e quelli meno significativi di ingresso; si considerino attivi a livello alto ed inoltre la richiesta di interruzione sia attivata per funzione OR a cui concorrono solamente i due bit di ingresso meno significativi. Le istruzioni del programma potrebbero essere:

```
LD     A, CFH            si predispone il modo di funzionamento
                         3: 1100 1111
```

```
OUT     (C3H),A
LD      A,0FH      linee D0 - D3: ingresso, D4 - 7: uscita
OUT     (C3H),A
LD      A,B7H      abilit. interruz. linee attive a livello alto,
                   funzione OR segue maschera 1101 0111
OUT     (C3H),A
LD      A,FCH      maschera per abilita solo D0 e D1
OUT     (C3H),A
```

Quanto è stato finora illustrato ha messo in evidenza solo le caratteristiche funzionali di una PIO: per ulteriori dettagli in particolare per quanto riguarda le caratteristiche statiche e dinamiche si rimanda ai manuali forniti dai costruttori.
Si vuole sottolineare la notevole semplicità di collegamento fra questo componente ed il microprocessore Z80: nel caso si volesse impiegarlo anche con altri processori si dovrebbe sicuramente predisporre una certa struttura hardware di interfacciamento con l'unità centrale, da mettersi a punto caso per caso, ed inoltre dovrebbe essere fatta una accurata verifica del comportamento dinamico, che richiede necessariamente la verifica dei legami temporali fra i diversi segnali di controllo.

6.7. L'8255

Altri dispositivi di interfacciamento parallelo presentano fondamentalmente le stesse caratteristiche di base viste nella PIO. Differiscono nei segnali di controllo in quanto sono stati progettati per funzionare con microprocessori diversi.
Si analizzano ora le caratteristiche del dispositivo 8255 realizzato per essere utilizzato con i microprocessori della linea INTEL, come ad esempio l'8080, 8085, ecc. Lo schema a blocchi di questo componente è riportato nella fig. 6.32.
Si hanno a disposizione 3 porte A, B e C per il trasferimento di dati, ognuna di 8 bit: in particolare i segnali sulla porta C possono assumere parecchie funzioni.
Il collegamento con l'unità centrale avviene, oltre che con il bus dei dati, con i segnali di controllo WR/ e RD/, il cui significato è ovvio, oltre a due segnali del bus degli indirizzi, A0 e A1, utilizzati praticamente per le stesse funzioni di B/A SEL e C/D SEL viste nella PIO. Infatti con una opportuna configurazione di questi due segnali si seleziona quella delle tre porte cui ci si riferisce mentre i due comandi WR/ e RD/ stabiliscono il tipo di operazione da eseguire: o di ingresso o di uscita. Una combinazione di tali segnali permette di informare l'8255 che quanto è presente sul bus dei dati deve essere interpretato come un comando mediante il quale si ha la possibilità di programmare i diversi modi di funzionamento di questo dispositivo.

D_7-D_0	DATA BUS (BI-DIRECTIONAL)
RESET	RESET INPUT
$\overline{CS}$	CHIP SELECT
$\overline{RD}$	READ INPUT
$\overline{WR}$	WRITE INPUT
A0, A1	PORT ADDRESS
PA7-PA0	PORT A (BIT)
PB7-PB0	PORT B (BIT)
PC7-PC0	PORT C (BIT)
V_{CC}	+5 VOLTS
GND	0 VOLTS

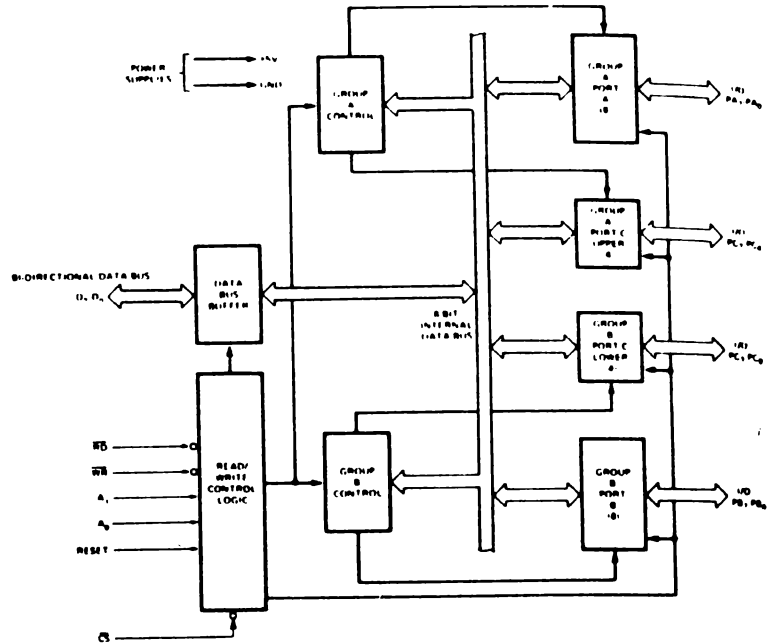



Fig. 6.32
Schema a blocchi dell'8255 (Intel Component Data Catalog 1979).

Esistono tre diversi modi di funzionamento le cui principali caratteristiche sono schematizzate nella fig. 6.33.
Nel modo 0 non si hanno a disposizione segnali di handshaking: si può ritenere che il dispositivo si comporti come un normale buffer. Ognuna delle porte può essere programmata o di ingresso o di uscita; in particolare la porta C può essere funzionalmente considerata divisa in due parti attribuibili da programma alla porta A o B, a seconda delle necessità.
Le uscite presentano la caratteristica di latch; tale caratteristica non è presente se si predispone la porta in input.
Nel modo 1 invece un certo numero delle linee della porta C sono utilizzate come segnali di handshaking e di richiesta di interruzione. Ognuna delle due porte A, e B può essere programmata sia di ingresso che di uscita con le linee in grado di memorizzare l'informazione. Non essendo prevista la

8255A BASIC OPERATION

A_1	A_0	$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	INPUT OPERATION (READ)
0	0	0	1	0	PORT A → DATA BUS
0	1	0	1	0	PORT B → DATA BUS
1	0	0	1	0	PORT C → DATA BUS
					OUTPUT OPERATION (WRITE)
0	0	1	0	0	DATA BUS → PORT A
0	1	1	0	0	DATA BUS → PORT B
1	0	1	0	0	DATA BUS → PORT C
1	1	1	0	0	DATA BUS → CONTROL
					DISABLE FUNCTION
X	X	X	X	1	DATA BUS → 3-STATE
1	1	0	1	0	ILLEGAL CONDITION
X	X	1	1	0	DATA BUS → 3-STATE

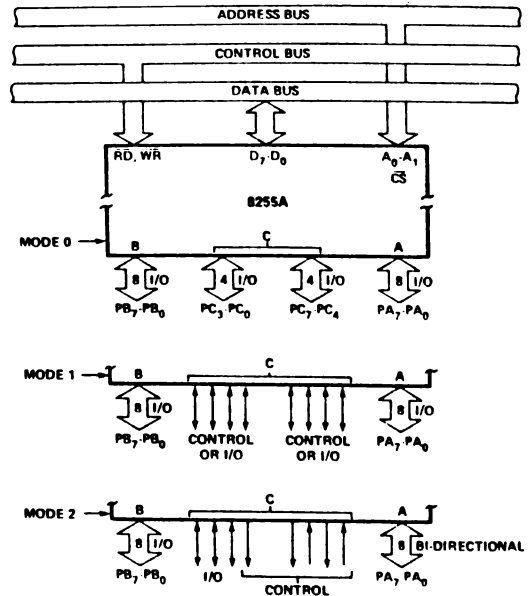



Fig. 6.33
Modi di funzionamento dell'8255 (Intel Component Data Catalog 1979)

vettorizzazione delle interruzioni, sono presenti due segnalazioni di interruzione, una per ogni porta; esiste inoltre la possibilità di abilitare o meno le richieste di interruzione. I segnali di handshaking assumono nomi diversi rispetto a quelli visti nella PIO, ma hanno la stessa funzione.

E' previsto un modo di funzionamento bidirezionale (modo 2) solo però per la porta A, che in questo caso presenta oltre che le uscite anche gli ingressi con caratteristiche latch. Quattro linee della porta C sono utilizzate per le segnalazioni di handshaking, mentre le restanti linee sono utilizzabili sia come linee di ingresso che di uscita. Non è previsto un funzionamento del tipo "control", come nella PIO modo 3, non tanto per la possibilità di usare alcune linee in ingresso ed altre in uscita quanto per il fatto che non sono presenti le diverse funzioni del modo 3 come l'attivazione a livello alto o basso, l'AND oppure l'OR per la richiesta delle interruzioni.

Capitolo 7

CONTATORI E TEMPORIZZATORI

7.1. Il controllo di intervalli di tempo

Molto spesso occorre fare eseguire un segmento di programma ad intervalli regolari di tempo, oppure dopo che sia trascorso un certo tempo dal verificarsi di un evento: si tratta in ogni caso di un controllo di intervalli temporali.

Si potrebbe ottemperare a queste specifiche solo tramite il software. E' possibile infatti organizzare una subroutine che ripeta un prefissato numero di istruzioni e mandarla in esecuzione tante volte quante sono necessarie per ottenere l'intervallo di tempo desiderato; tale routine di ritardo potrebbe essere organizzata come indicatò nel diagrâmma di flusso di fig. 7.1.

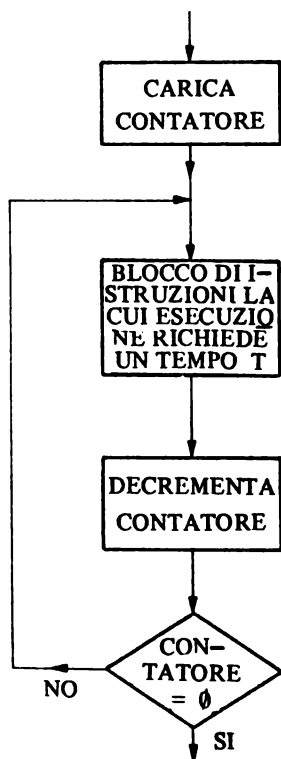



Fig. 7.1
Diagramma di flusso di una routine di ritardo.

Predisponendo ad un valore opportuno il contatore e conoscendo il tempo impiegato per l'esecuzione del blocco delle istruzioni, si può ottenere qualunque intervallo di tempo si desideri, eventualmente ricorrendo a contatori multipli.
La subroutine può essere organizzata in forma parametrica così da poterla utilizzare qualsiasi sia il caso che si presenta; con tale soluzione si hanno notevoli vantaggi soprattutto nei confronti della flessibilità: basta infatti variare solo il contenuto del contatore per ottenere qualsiasi intervallo di tempo richiesto.
Tale soluzione presenta però anche degli svantaggi che, in qualche applicazione, possono addirittura essere non accettabili.
Un primo inconveniente è la scarsa utilizzazione del microprocessore: infatti l'unità centrale rimane completamente impegnata per l'esecuzione della routine di ritardo.
Non è però questo l'inconveniente più grave: si consideri ad esempio il caso di un programma per il controllo di un processo in cui le periferiche siano in grado di effettuare richieste di interruzione. Se durante l'esecuzione della routine di ritardo è presentata alla CPU una richiesta di interruzione, e questa è servita, il ritardo introdotto dalla routine non è più quello voluto. Infatti esso dipende ora anche dal tempo impiegato per l'esecuzione delle routine di servizio dell'interruzione, il cui valore è in genere non prevedibile dato che dipende dallo stato e dal tipo di intervento che le diverse periferiche possono richiedere.
Una soluzione a tale inconveniente potrebbe essere quella di disabilitare il processore ad accettare richieste di interruzione durante l'esecuzione della routine di ritardo. In questo caso è mantenuta la precisione del ritardo, ma si pospone il servizio delle interruzioni e ciò non sempre è possibile. Da quanto detto si può concludere che l'utilizzo del software, per generare dei ritardi presenta delle caratteristiche non sempre accettabili.
Occorre allora passare a soluzioni di tipo hardware, le quali possono essere di tipo diverso a seconda dell'entità del ritardo che si desidera ottenere e della precisione richiesta.
Si può ad esempio utilizzare un contatore, al cui ingresso è inviato il segnale di clock del sistema, o altro segnale a frequenza costante; tale contatore fornisce una segnalazione, di solito una richiesta di interruzione, alla fine del conteggio.
E' evidente la notevole riduzione d'impegno dell'unità centrale, la quale ora deve intervenire solo alla fine del conteggio per intraprendere l'elaborazione prevista. Con questa semplice soluzione non si ha però la possibilità di variare l'entità del ritardo in quanto per far ciò occorre modificare il contatore.
Con una diversa struttura hardware è però possibile ottenere questa flessibilità: uno schema a blocchi che potrebbe essere utilizzato è riportato nella fig. 7.2.

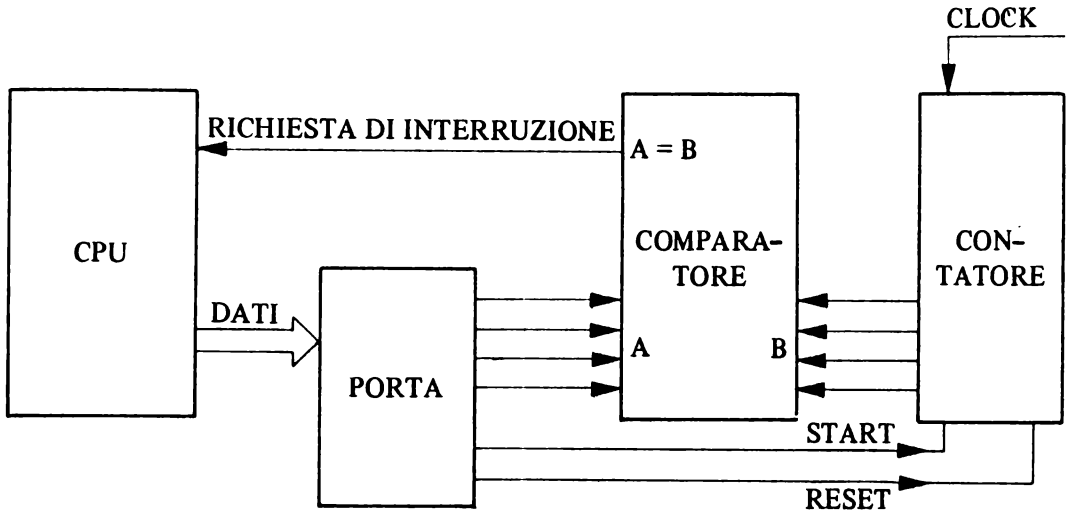



Fig. 7.2
Schema a blocchi di un circuito che genera una richiesta di interruzione dopo che è trascorso un intervallo di tipo programmato.

I segnali di uscita del contatore sono collegati all'ingresso A di un comparatore; l'altro ingresso B è posto ad un valore prefissato dall'unità centrale tramite l'invio di un dato su di una porta di uscita. Quando l'uscita del contatore assume la configurazione uguale a quella presente sulla porta di uscita, il comparatore attiva la sua uscita "A=B" e quindi invia una richiesta di interruzione all'unità centrale per segnalare che è passato il tempo che era stato impostato in precedenza. I segnali di azzeramento e di abilitazione del contatore debbono essere ovviamente inviati dal microprocessore, come indicato nello schema di fig. 7.2.
Per risolvere esigenze di questo tipo si possono utilizzare componenti predisposti per fornire queste specifiche funzioni; ciò, come si vedrà, permette una notevole semplificazione sia dell'hardware che del software dell'intero sistema di elaborazione.
Un problema abbastanza simile all'introduzione di un ritardo è quello del conteggio di eventi esterni: in questo caso il microprocessore deve intervenire solo dopo che un certo evento si è verificato un prefissato numero di volte.
Allo scopo è necessaria la presenza di un hardware esterno in quanto si

deve segnalare il verificarsi dell'evento, che generalmente è asincrono nei riguardi dell'unità centrale.
La soluzione più semplice ed intuitiva che si può adottare consiste nel segnalare all'unità centrale il verificarsi di ogni evento mediante una richiesta di interruzione. La relativa routine di servizio ogni volta che è mandata in esecuzione incrementa un contatore, realizzato via software. ne confronta il contenuto con il numero di eventi che si vuole avvengano e solo quando ne verifica l'uguaglianza dei valori manda in esecuzione l'elaborazione richiesta.
Anche in questo caso però può essere conveniente adottare delle soluzioni hardware, come quella indicata in fig. 7.2, nella quale questa volta il contatore riceve un impulso di clock al verificarsi di ogni evento. Si può notare che la stessa struttura hardware può consentire di risolvere i due problemi di introduzione di un ritardo e di conteggio di eventi con semplici variazioni circuitali.
E' per questo motivo che la maggior parte dei dispositivi programmabili che servono allo scopo sono stati realizzati in modo da poter essere impiegati nelle due situazioni ed essi infatti prendono in genere il nome di "contatori/temporizzatori". La loro utilizzazione è prevista di solito per un particolare processore dal quale ricevono i vari segnali di controllo, così che se si usano unità centrali diverse può essere necessario predisporre dei circuiti di adattamento. Si illustrano ora le caratteristiche principali del contatore-temporizzatore realizzato per essere utilizzato con il processore Z80.

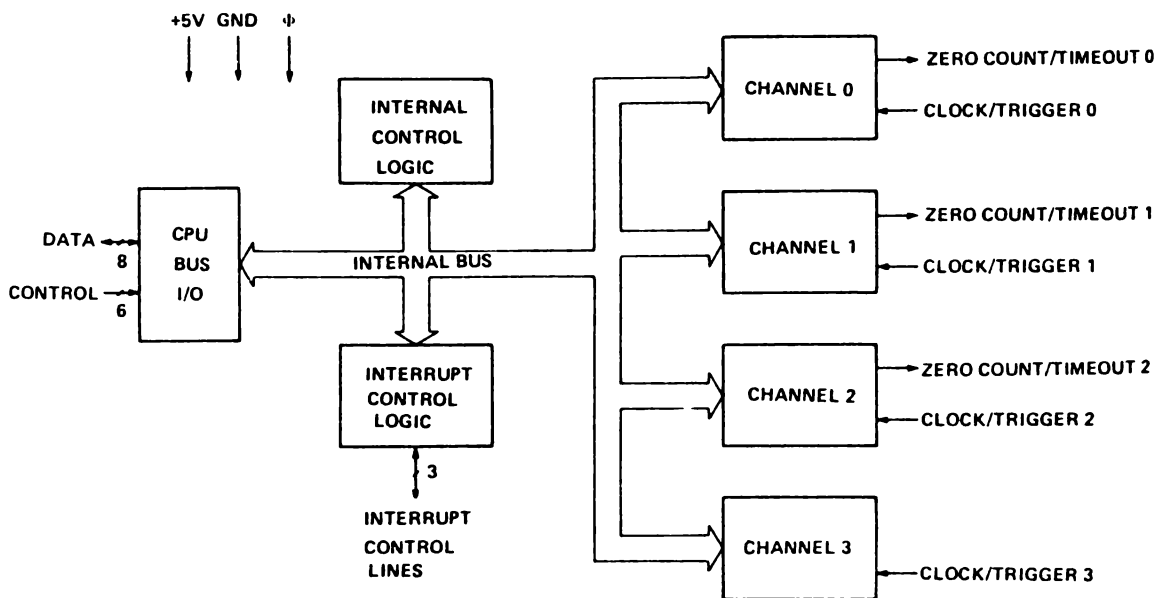



Fig. 7.3
Schema a blocchi del CTC (Mostek).

7.2. Il CTC (Counter Timer Circuit) per lo Z80

Il CTC è un componente programmabile, che presenta quattro canali indipendenti di conteggio, i quali possono essere utilizzati sia come contatempi che come contaeventi, con diverse modalità di funzionamento.
Uno schema a blocchi del CTC è riportato in fig. 7.3.
Ad ognuno dei quattro canali sono associati dei segnali di ingresso e di uscita che assumono significati diversi a seconda del modo di funzionamento che è stato programmato per il canale.
Sono presenti nel dispositivo un blocco di controllo, che coordina il funzionamento complessivo del componente, una interfaccia per il collegamento con l'unità centrale, ed un blocco per la gestione delle interruzioni, la cui struttura è identica a quella vista nel caso della PIO.
I vari segnali che interessano tale componente sono riportati nella fig. 7.4.
I dati ed i comandi relativi al CTC sono trasferiti attraverso le otto linee del bus dei dati e sono qualificati dai valori assunti dai segnali di controllo; fra questi si trovano M1/, IORQ/ e RD/, con lo stesso significato visto nella PIO: tali comandi sono generati dallo Z80 e permettono sia di sincronizzare il CTC con la CPU sia di indicare che è in svolgimento una operazione di lettura o di scrittura oppure di accettazione delle interruzioni con le modalità viste a suo tempo. Esiste inoltre un ingresso di abilitazione CS/ per la selezione del CTC durante il trasferimento delle informazioni con l'unità centrale. E' anche presente un ingresso di RESET mediante il quale si predispone in prefissate condizioni iniziali il CTC.
Per la selezione di uno dei quattro canali sono utilizzati i due ingressi CS0 e CS1, secondo la tabella 7.1.

Tabella 7.1

CS1	CS0	
0	0	è selezionato il canale 0
0	1	è selezionato il canale 1
1	0	è selezionato il canale 2
1	1	è selezionato il canale 3

Di solito a questi due ingressi sono inviate le due linee meno significative del bus degli indirizzi: uno schema di collegamento potrebbe essere quello indicato in fig. 7.5.
Con i collegamenti indicati il CTC è selezionato per tutti gli indirizzi che presentano i sei bit più significativi nella configurazione 100111, per cui i quattro canali sono individuati dagli indirizzi:

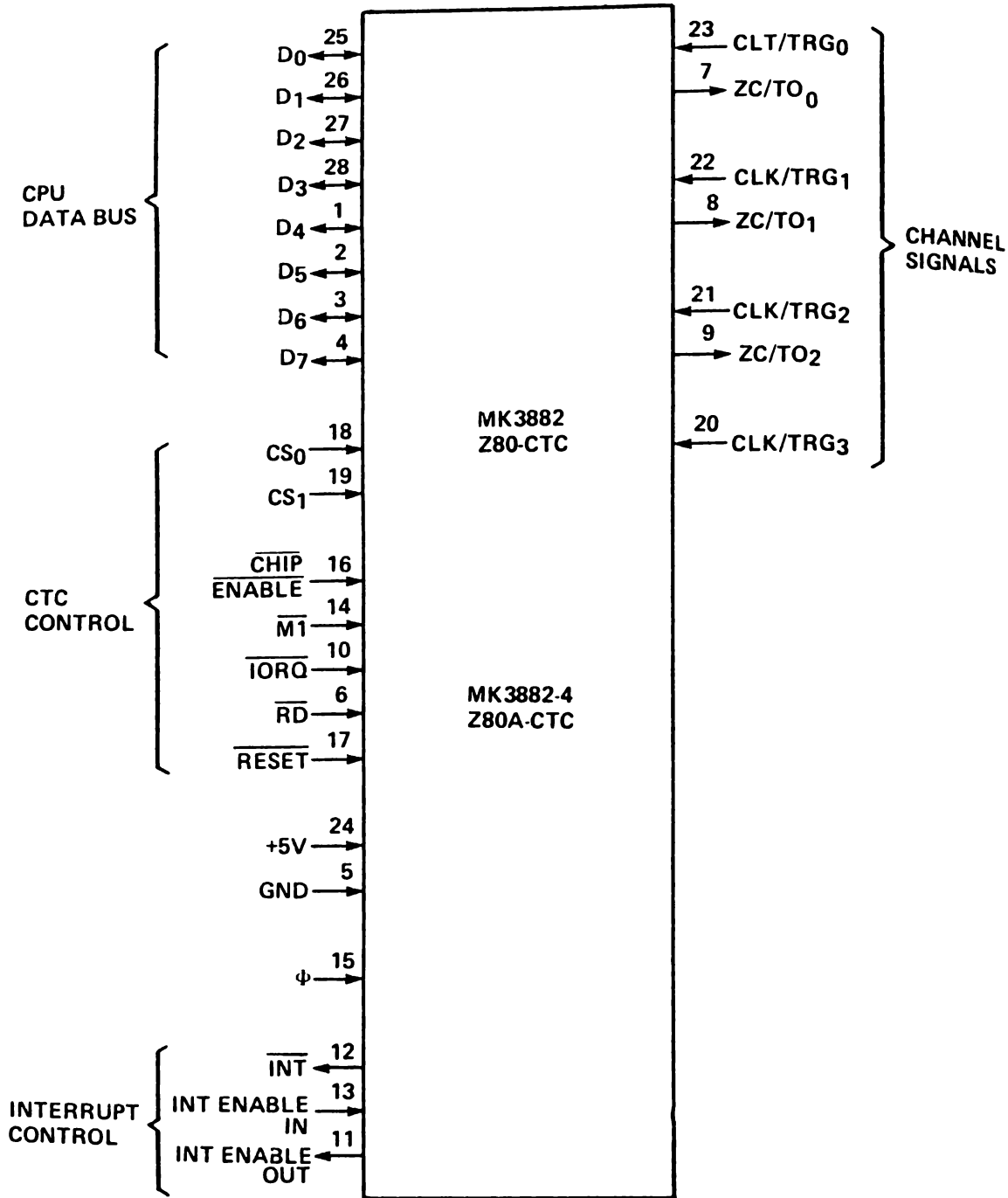



Fig. 7.4.
I vari segnali del CTC (Mostek).

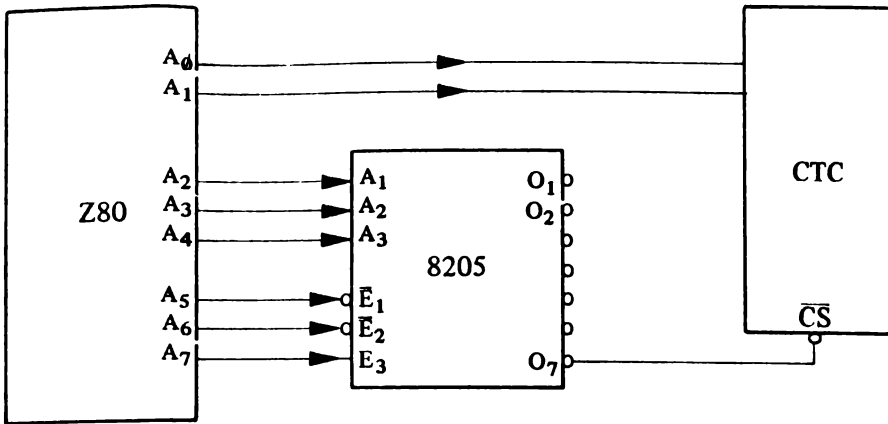



Fig. 7.5
Collegamento del CTC allo Z80.

9CH canale 0
9DH canale 1
9EH canale 2
9FH canale 3

Nella fig. 7.4 sono anche indicati i segnali per la gestione delle interruzioni, che sono identici a quelli visti per il componente PIO. E' inoltre presente un ingresso di clock ϕ che serve sia per il controllo dei ritardi nel modo timer, sia per la sincronizzazione del funzionamento del CTC con lo Z80.
Esistono poi degli ingressi, che permettono di attivare il funzionamento dei diversi canali, e delle uscite che danno conto del verificarsi di particolari condizioni all'interno dei canali stessi.
In particolare i quattro ingressi CLK/TRG, uno per ogni canale, sono utilizzati come segnali di clock esterno o come segnali di trigger.
Solamente per i primi tre canali è inoltre presente un segnale di uscita che assume il valor logico alto ogni volta che il relativo contatore raggiunge il valore zero.
Lo schema a blocchi di uno dei canali è riportato nella fig. 7.6.

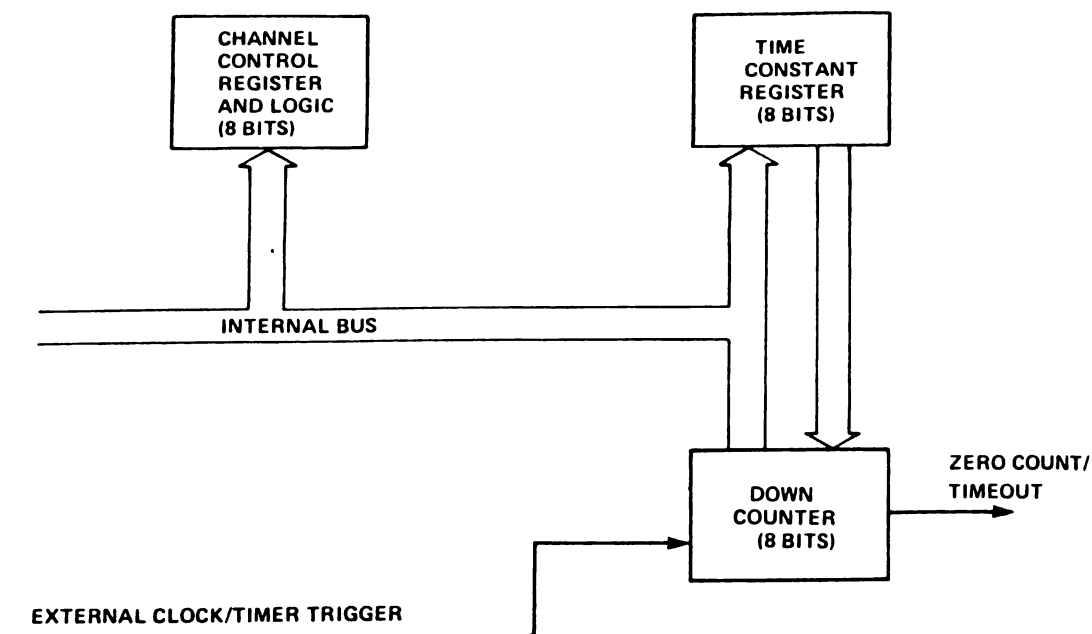



Fig. 7.6
Schema a blocchi di uno dei canali del CTC (Mostek).

Ogni canale è dotato di una rete logica e di un registro di controllo il cui compito è di memorizzare le informazioni inviate dalla CPU, atte a stabilirne le modalità di funzionamento. Per quanto riguarda queste ultime, ne esistono due: il modo contaeventi (counter) e quello contatempi (timer). Nel primo il CTC conta gli impulsi che si presentano all'ingresso CLK/TRG di ogni canale. Con una opportuna parola di comando è possibile programmare il CTC in modo che l'aggiornamento del contatore avvenga sul fronte di salita, oppure su quello di discesa, dell'impulso inviato dall'esterno. Il contatore indicato in fig. 7.6 è un contatore all'indietro: quando il suo contenuto, per decrementi successivi raggiunge lo zero, attiva l'uscita ZC/TO (presente solamente nei primi tre canali); contemporaneamente inoltre il contatore riassume il valore iniziale, programmato precedentemente dalla CPU e mantenuto immagazzinato nel registro, "Time Constant Register", presente in ogni canale. Con questa organizzazione il conteggio è quindi ripetuto con modulo pari al valore impostato inizialmente.
E' anche ammesso che la CPU modifichi il valore iniziale durante il con-

teggio: in questo caso il CTC utilizzerà il nuovo valore del modulo solo dopo aver raggiunto il valore zero relativo al conteggio precedente.
Il CTC, una volta raggiunto lo zero, può attivare una richiesta di interruzione se a ciò è abilitato. Il valore iniziale massimo che può essere attribuito al contatore è 255 in quanto il registro a disposizione è di 8 bit. Qualora siano necessari dei conteggi più grandi si può organizzare un adatto programma, come detto in precedenza, oppure si possono anche collegare in cascata più canali: utilizzandone solo due si può ottenere un conteggio massimo di 256 x 256 = 65536. Se si vogliono contare ad esempio 10.000 eventi si può organizzare il circuito indicato in fig. 7.7.

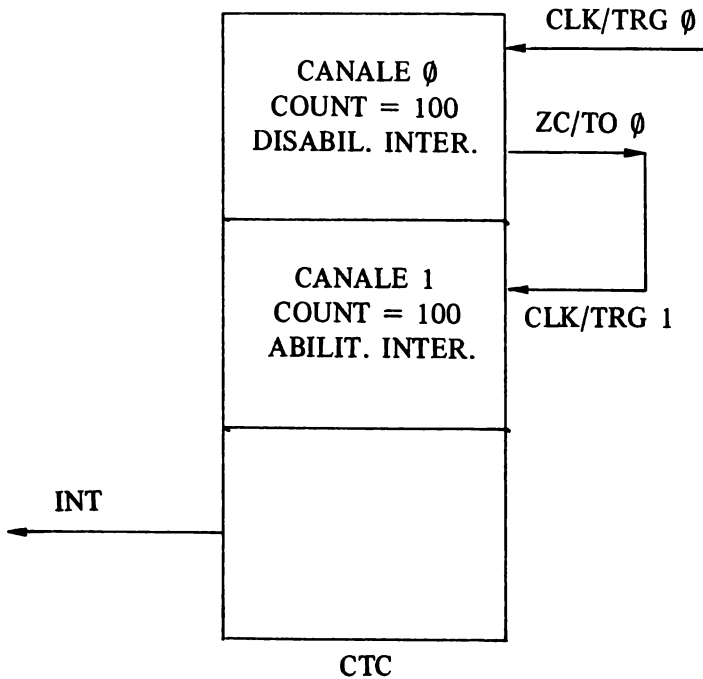



Fig. 7.7
Due canali del CTC connessi in cascata per conteggiare 10.000 eventi.

Entrambi i contatori dei canali 0 e 1 sono caricati con il valore iniziale 100 e l'uscita del canale 0 è collegata all'ingresso del canale 1; inoltre il canale 0 non deve essere stato abilitato a fare richieste di interruzione, contrariamente al canale 1. Con questa organizzazione si ha una richiesta di interruzione ogni 10.000 eventi esterni.
Nel modo di funzionamento contatempi il segnale di riferimento è il clock ϕ del sistema, utilizzato anche per l'unità centrale, e che quindi ha una frequenza di 2.5 oppure 4 MHz a seconda del tipo di processore Z80 che si utilizza. La definizione temporale che in questo modo si riesce ad otte-

nere è molto elevata, ed in genere non richiesta nelle normali applicazioni, dato che il periodo di clock risulta rispettivamente di 400 e 250 ns. Inoltre per ottenere intervalli di tempo di una durata dell'ordine dei millisecondi sarebbe necessario un numero superiore di bit, rispetto a quelli presenti nel contatore all'indietro. Per ovviare a questo inconveniente è presente un contatore-divisore (prescaler) il quale invia un impulso al contatore di conteggio solo ogni 16, oppure ogni 256, periodi del clock ϕ; la scelta fra queste due possibilità può essere fatta da programma. Anche in questo modo di funzionamento il raggiungimento del valore zero da parte del contatore all'indietro provoca l'attivazione di ZC/TO, la richiesta di interruzione, se il CTC è stato precedentemente abilitato, ed il ripristino del valore iniziale nel contatore.
All'uscita della linea ZC/TO è presente quindi una sequenza di impulsi la cui frequenza può variare da un minimo ad un massimo a seconda del valore impostato nel contatore e selezionato nel prescaler. In termini temporali il valore minimo si ottiene quando il contatore è stato inizialmente caricato ad uno ed il prescaler selezionato per dividere per 16. A seconda del periodo di clock, il periodo minimo ottenibile risulta:

$250 \times 10^{-9} \times 16 = 4\ \mu s$ con periodo di clock di 250 ns

$400 \times 10^{-9} \times 16 = 6{,}4\ \mu s$ con periodo di clock di 400 ns.

Invece il periodo massimo si avrà quando il contatore è stato inizialmente caricato con 0, che corrisponde a 256 conteggi, ed il prescaler è stato predisposto a 256. Si ottiene, nei due casi:

$$250 \times 10^{-9} \times 256 \times 256 = 16{,}384 \text{ ms}$$

$$400 \times 10^{-9} \times 256 \times 256 = 26{,}2 \text{ ms}$$

Se si utilizzassero i quattro canali in cascata, con il prescaler del primo canale ed i quattro contatori tutti caricati con valore iniziale pari a 256, come indicato nella fig. 7.8, si avrebbe una richiesta di interruzione, con un clock di 400 ns, pari a:

$$400 \times 10^{-9} \times 256^5 = 4.398 \times 10^5 \text{s} = 122.000 \text{ ore}$$

ed è quindi possibile ottenere in pratica qualsiasi intervallo di tempo occorra.

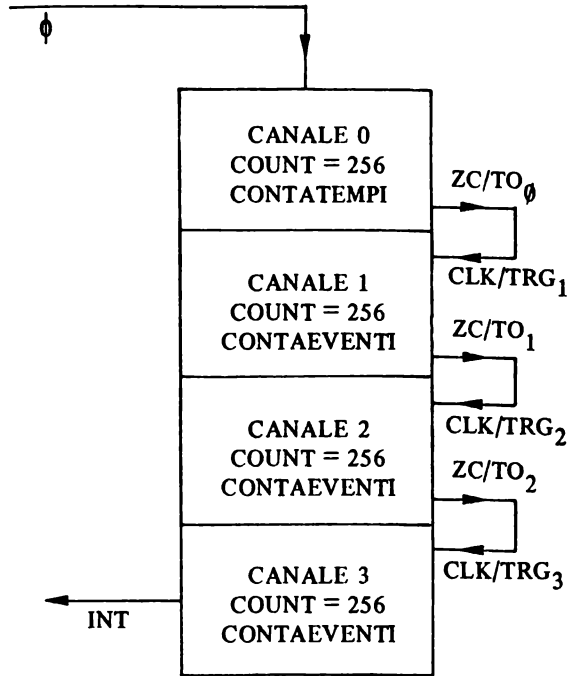



Fig. 7.8
I quattro canali del CTC connessi in cascata.

7.3. Programmazione del CTC

Per programmare il CTC si devono inviare ad ogni canale dei comandi, formati da alcuni byte, che permettono di selezionare le diverse possibilità di funzionamento.
Per quanto riguarda il modo ed i parametri di un canale si deve inviare un byte il cui formato è indicato nella fig. 7.9.

D7 ... D0

ABILITAZIONE INTER.	MODE	RANGE	SLOPE	TRIGGER	LOAD	RESET	1

Fig. 7.9
Il byte di controllo di modo per un canale del CTC.

Il significato associato ai vari bit è descritto nella lista che segue:

D7: ponendo a 1 tale bit si abilita il canale a richiedere interruzione quando il corrispondente contatore all'indietro ha raggiunto il valore 0; in caso contrario il CTC non fa richiesta di interruzione. Si ricorda, che

il CTC, al pari delle altre periferiche della famiglia Z80 è predisposto per gestire le interruzioni con l'unità centrale funzionante nel modo 2.

D6: se tale bit è posto ad 1 il canale funziona come contaeventi, in caso contrario come contatempi.

D5: questo bit ha significato solamente per il modo di funzionamento come contatempi; se posto a 1 il prescaler è programmato a 256, altrimenti a 16.

D4: permette di stabilire il fronte che attiva una determinata azione; se posto ad 1, nel modo contaeventi, il contatore sarà decrementato sul fronte di salita del segnale CLT/TRG; se invece il canale funziona in modo contatempi questo bit ha significato solamente se l'inizio del conteggio è comandato da un segnale esterno di trigger: anche in questo caso, se il D4 è posto a 1 il conteggio inizia sul fronte positivo del segnale di trigger. I fronti attivi saranno invece quelli di discesa se tale bit è programmato a 0.

D3: anche questo bit serve solamente se il canale funziona come contatempi: se posto al valore 1 la partenza del conteggio avviene, su segnale esterno, sul fronte stabilito dal valore del bit D4.

D2: indica, quando è posto al valore 1, che il successivo byte, inviato a quello stesso canale, deve essere interpretato come il valore da caricare nel registro per il conteggio.

D1: se tale bit assume il valore 0 il canale arresta l'operazione di conteggio senza tuttavia alterare il contenuto dei suoi registri interni. Questi possono essere alterati inviando i dati da memorizzare preceduti dagli adatti comandi. Per fare ripartire il conteggio è necessario l'invio di questa stessa parola di comando con i due bit D2 e D1 entrambi posti a 1.

D0: tale bit deve essere necessariamente posto a 1.

Nel caso che con la parola di controllo di fig. 7.9 il CTC sia stato abilitato a fare delle richieste di interruzione, deve essergli inviato come ulteriore byte di programmazione anche il vettore per la individuazione dell'inizio della routine di servizio. Tale vettore deve essere inviato al canale 0, qualunque sia il canale interessato, e deve avere il formato indicato in fig. 7.10.
Il bit D0 deve essere necessariamente 0 mentre non sono presi dal CTC in considerazione i bit D1 e D2, che quindi possono assumere qualsiasi valore. Quando la CPU richiede il vettore a seguito di una interruzione, il CTC pone sul bus dei dati il valore dei bit D7, D6, D5, D4, D3 e D0 precedentemente inviati mentre i bit D2 e D1 assumono un valore che dipende dal canale che ha generato la richiesta di interruzione. Di questo deve essere

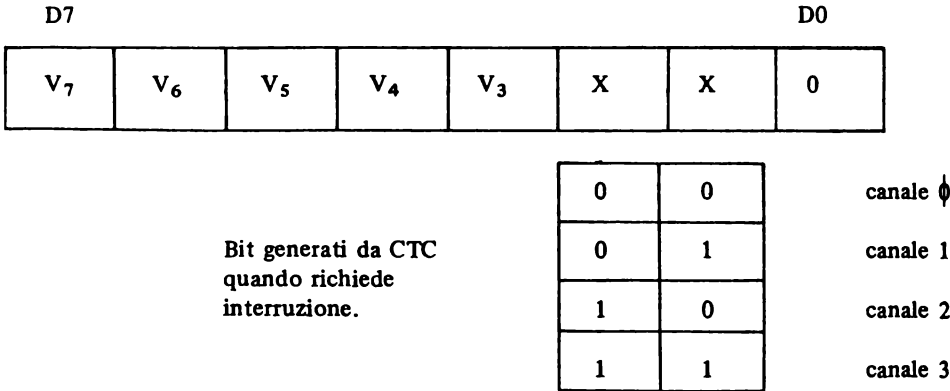



Fig. 7.10
Formato del settore di interruzione.

tenuto conto per l'organizzazione della tabella da cui ricavare l'indirizzo di partenza della routine di servizio delle interruzioni. Se più canali fanno richiesta contemporaneamente è stabilita una priorità interna: il canale 0 ha il livello più elevato mentre il 3 quello più basso.
Si da ora un esempio di programmazione del CTC.
Si supponga di utilizzare lo schema di collegamento, per la individuazione dei canali, riportato nella fig. 7.5; inoltre la tabella della routine di interruzione abbia inizio dall'indirizzo simbolico TAB che si suppone sia pari.
Si vuole programmare il canale 2 come contaeventi per dare una segnalazione di interruzione ogni 8 eventi. Il programma di inizializzazione può essere così organizzato:

IM2		si predispone lo Z80 a funzionare in modo 2
LD	HL, TAB	
LD	A,H	carica il Reg. I con la parte più significativa
LD	I, A	dell'indirizzo TAB
LD	IX, ROUT2	in IX indirizzo di partenza della Routine di servizio per il canale 2
LD	(TAB+04), IX	posiziona indirizzo in TAB per canale 2
LD	A, (LSB TAB)	in A si carica il Byte meno significativo dell'indirizzo TAB (D0 = 0)
OUT	(9 CH), A	manda il vettore al canale 0
LD	A, 0C5H	si invia il byte 11000101 per abilitare le richieste di interruzione, fissare il modo contaeventi - il conteggio avviene sul fronte di discesa e seguirà una costante

```
OUT   (9EH),A          si carica il canale
LD    A,8
OUT   (9EH),A          si carica il contatore a 8.
```

Per ottenere il funzionamento come contatempo si deve cambiare la configurazione del byte di comando secondo quanto desiderato ed inviare inoltre le informazioni che riguardano il valore del prescaler e il modulo del conteggio.
Il CTC si presta anche ad un'altra interessante applicazione che permette ad una periferica qualsiasi di utilizzare il modo 2 di gestione delle interruzioni, senza che occorra realizzare la struttura hardware richiesta. Si può infatti ricorrere all'impiego di un CTC secondo quanto indicato nello schema a blocchi di fig. 7.11.

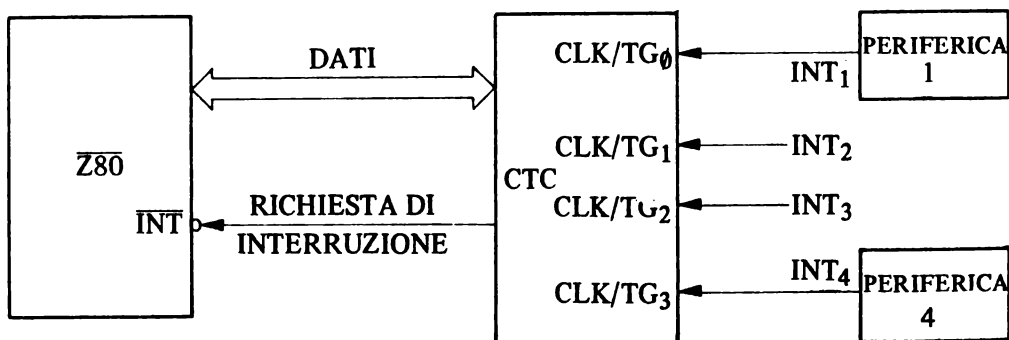



Fig. 7.11
Il CTC come controllore di interruzioni.

Ogni canale deve essere programmato per funzionare nel modo contaeventi e abilitato a fare richieste di interruzione; inoltre ad ogni contatore deve essere stato inviato 1 come valore iniziale per cui, ogni volta che è attivato dalla periferica l'ingresso corrispondente, si ha la generazione di una richiesta di interruzione. Il CTC fornisce alla CPU il vettore richiesto, che individua, da quanto detto, la periferica che richiede l'interruzione. Ogni CTC è in grado perciò di gestire 4 sorgenti esterne risolvendo sia la priorità in-

terna che quella riguardante altre sorgenti, data la possibilità di realizzare il daisy-chain.

7.4. Il contatore-temporizzatore 8253

Esistono altri dispositivi che permettono di ottenere le funzioni fin qui analizzate; uno di questi è l'8253, che può essere utilizzato con i microprocessori 8080, 8085, ecc. Lo schema a blocchi è riportato nella fig. 7.12.

D_7 D_0	DATA BUS (8 BIT)
CLK N	COUNTER CLOCK INPUTS
GATE N	COUNTER GATE INPUTS
OUT N	COUNTER OUTPUTS
$\overline{RD}$	READ COUNTER
$\overline{WR}$	WRITE COMMAND OR DATA
$\overline{CS}$	CHIP SELECT
A_0 A_1	COUNTER SELECT
V_{CC}	+5 VOLTS
GND	GROUND

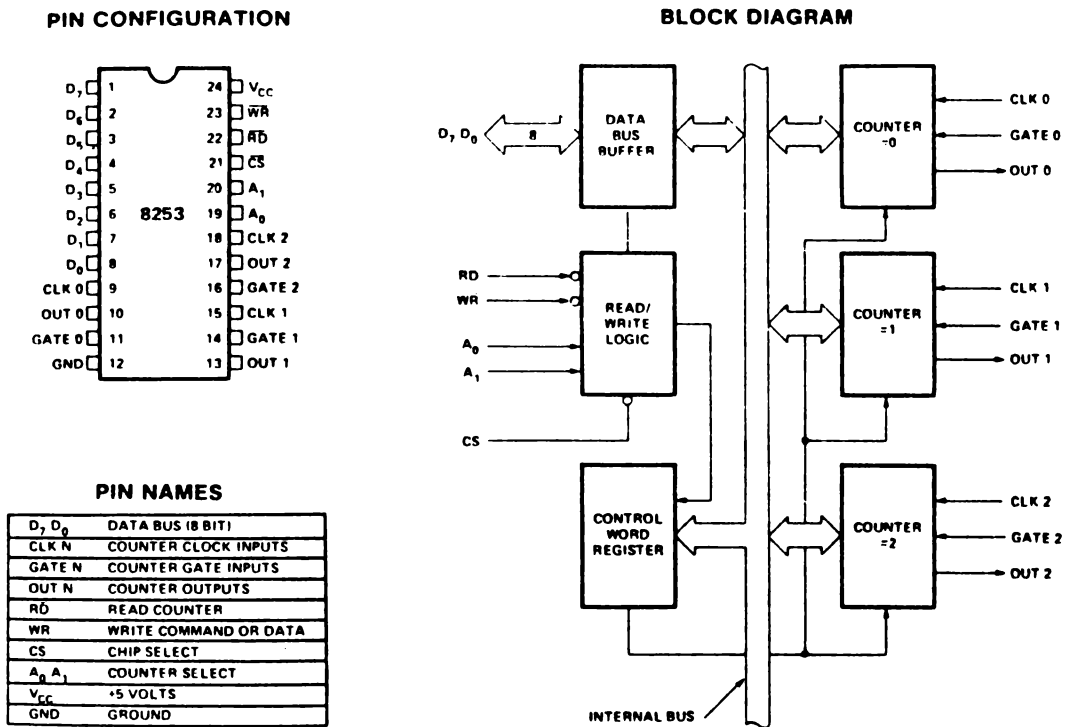



Fig. 7.12
Schema a blocchi dell'8253. (Intel, Component Data. Catalog 1979).

Come si può notare sono presenti tre soli canali di conteggio ognuno dei quali presenta dei segnali di ingresso e uscita che permettono una notevole varietà di modi di funzionamento.

Il collegamento con l'unità centrale avviene tramite il bus dei dati ed alcune linee di controllo; sono presenti quattro di queste linee oltre, naturalmente, al segnale di CS/. Con una opportuna combinazione assunta agli ingressi A_0 ed A_1 è possibile accedere direttamente ad uno dei canali sia per un trasferimento di dati sia per inviare comandi. Il valore che devono assumere i vari segnali per ottenere determinate attività è riportato nella tab. 7.2.

Tabella 7.2

$\overline{CS}$	$\overline{RD}$	$\overline{WR}$	A_1	A_0	
0	1	0	0	0	Load Counter No. 0
0	1	0	0	1	Load Counter No. 1
0	1	0	1	0	Load Counter No. 2
0	1	0	1	1	Write Mode Word
0	0	1	0	0	Read Counter No. 0
0	0	1	0	1	Read Counter No. 1
0	0	1	1	0	Read Counter No. 2
0	0	1	1	1	No-Operation 3-State
1	X	X	X	X	Disable 3-State
0	1	1	X	X	No-Operation 3-State

Valori dei segnali per il comando dell'8253. (Intel, Component Data. Catalogo 1979).

Ogni canale possiede un contatore interno di 16 bit per cui il conteggio in questo caso può iniziare da 65535 e non solo da 256 come nel CTC.
I vari canali possono operare in cinque modi distinti e ognuno ha un suo proprio ingresso di clock mediante il quale è immesso il segnale che deve essere preso in considerazione per il conteggio. Non è detto ovviamente che questo ingresso debba necessariamente essere un segnale periodico: potrebbe anche essere il segnale di ingresso per la funzione contaeventi. La disponibilità di ingressi distinti anche nel caso di funzione contatempi, se da un lato può rendere più complessa la struttura hardware esterna, d'altra parte permette di utilizzare segnali di riferimento diversi e quindi aumenta la possibilità di applicazione.
Il segnale di uscita OUT fornisce la segnalazione di quando il contatore interno, che è anche in questo componente del tipo all'indietro, raggiunge lo zero.
Il conteggio può essere abilitato o meno dal segnale di ingresso GATE, la cui funzione può essere diversa a seconda del modo di funzionamento selezionato.

Il modo di funzionamento che più si avvicina a quello del CTC è il modo 0; in fig. 7.13 è riportato il diagramma temporale dei segnali nel caso di un canale funzionante in modo 0.

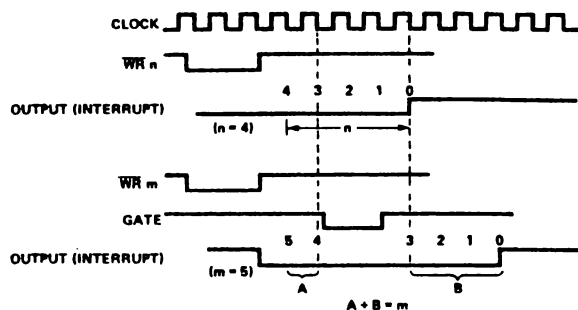



Fig. 7.13
Diagramma temporale relativo ad un canale dell'8253 funzionante in modo 0. (Intel, Component Data. Catalog 1979).

L'uscita OUT, inizialmente a livello basso, è resa attiva, al livello logico alto, una volta che il contatore sia stato caricato a che siano avvenuti il numero di eventi stabiliti e quindi può essere collegata alla richiesta di interruzioni dell'unità centrale. Non è prevista in questo caso una vettorizzazione ma ogni canale presenta il suo segnale di OUT per le richieste di interruzione. L'utilizzazione dell'ingresso GATE consente di arrestare il conteggio e farlo riprendere su comando esterno. Gli altri modi di funzionamento sono alquanto diversi da quelli visti nel caso del CTC e permettono di ottenere un segnale di uscita da OUT che è funzione sia del modo nel quale è stato programmato il dispositivo sia del valore assunto dal segnale GATE.
Il diagramma temporale del modo 1 è riportato nella fig. 7.14 dove si vede che il segnale di OUT si mantiene al livello basso solamente per un programmato numero di periodi di clock, una volta attivato il segnale GATE indicato in questo caso con il nome di TRIGGER.
Si noti che è possibile arrestare e far ripetere il conteggio se si disattiva e poi si riattiva il comando TRIGGER.
Il modo 3 invece permette di ottenere un generatore di onda quadra, come indicato nella fig. 7.15, dove sono riportate due diverse forme di onda quadra generate a seconda che il numero dei periodi di clock programmati sia pari oppure dispari.

MODE 1: Programmable One-Shot

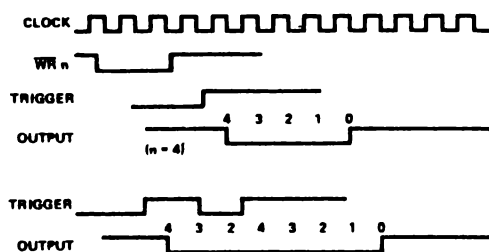



Fig. 7.14
Diagramma temporale relativo ad un canale dell'8253 funzionante in modo 1.
(Intel, Component Data. Catalog 1979).

MODE 3: Square Wave Generator

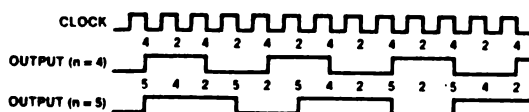



Fig. 7.15
Diagramma temporale relativo ad un canale dell'8253 funzionante in modo 3.
(Intel, Component Data. Catalog 1979).

Capitolo 8
I/O SERIALI

8.1. Trasmissione seriale

In un elaboratore è frequente la necessità di dover scambiare informazioni con dispositivi posti a distanze notevoli: in questo caso non è conveniente trasmettere i dati in forma parallela poiché occorrerebbe un elevato numero di linee per il collegamento; esistono d'altro canto delle periferiche che non accettano tale tipo di collegamento, come ad esempio la telescrivente. Si ricorre allora alla trasmissione seriale anche se essa è in genere più lenta rispetto a quella in parallelo.
In una trasmissione seriale si deve tener conto delle proprietà del canale di comunicazione, che può introdurre dei disturbi, e quindi è necessario utilizzare delle tecniche che permettano, se non la correzione degli errori introdotti, almeno di rivelarli.
E' perciò necessario introdurre delle ulteriori informazioni, oltre ai dati da trasmettere, per garantire una corretta ricezione dei dati stessi.
Le diverse formalità con cui si inviano queste informazioni aggiuntive distinguono i vari tipi di trasmissioni seriali attualmente utilizzati.
All'inizio di una qualsiasi trasmissione seriale è necessario inviare una segnalazione opportuna per informare il ricevitore dell'imminente arrivo di nuovi dati: questa segnalazione prende il nome di sincronismo. A seconda delle modalità con cui quest'ultimo è inviato si possono distinguere i due tipi fondamentali di trasmissioni seriali: quelle asincrone e quelle sincrone.
Ciò che le rende differenti consiste nel fatto che mentre la trasmissione asincrona richiede che l'informazione di sincronismo sia fornita per ogni dato trasmesso, nella sincrona questa informazione è necessario sia trasmessa solamente una volta all'inizio di un blocco composto da un certo numero di dati.
Il formato di una trasmissione asincrona è riportato nella fig. 8.1.
Normalmente la linea si trova ad un livello logico alto; l'inizio della trasmissione di un dato è individuato da un livello logico basso, che è detto bit di START. Seguono poi altri bit, che costituiscono il dato, in genere

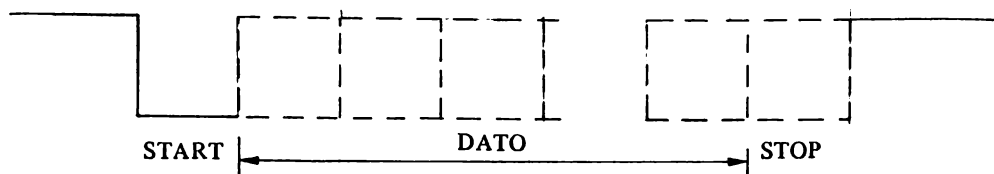



Fig. 8.1
Formato di una trasmissione asincrona.

in numero da 5 a 8, a seconda del tipo di codice adottato; la fine della trasmissione è indicata dalla presenza di uno o più bit di STOP, caratterizzati dal porre la linea a livello logico alto.
Il bit di START serve per informare il ricevitore di sincronizzare l'analisi dei valori logici presenti sulla linea in quanto ha inizio la trasmissione; il numero di bit che costituiscono un dato e la durata di ogni bit, cioè la velocità di trasferimento, devono essere stati precedentemente concordati fra il trasmettitore ed il ricevitore.
Per quanto riguarda la trasmissione sincrona, i dati, o caratteri, da trasmettere sono raggruppati in blocchi, o record, di dimensioni prefissate. Questi caratteri sono preceduti da uno, o a volte due, caratteri di sincronismo, i quali hanno ancora la funzione di informare il ricevitore che ha inizio una nuova trasmissione; questi caratteri di sincronismo servono anche per separare e distinguere un record da quello successivo.
Nelle trasmissioni seriali si possono usare varie tecniche per il controllo della correttezza della trasmissione; esse variano sia a seconda del tipo di trasmissione adottato, sia dal grado di sicurezza con cui si vuole garantire la correttezza dei dati in arrivo. Per quest'ultimo aspetto si fa notare come esso sia strettamente legato alle caratteristiche del canale di trasmissione utilizzato, come ad esempio la lunghezza della linea, la presenza di sorgenti di rumore ecc.
Una tecnica molto usata, in particolare in trasmissioni asincrone, consiste nell'aggiungere un bit di parità per ogni dato trasmesso. Questo bit assume il valore 0 o 1 in modo che complessivamente il numero di bit 1 trasmessi sia pari (oppure dispari) a seconda di quanto convenuto. Tale bit deve essere inserito dal trasmettitore e controllato dal ricevitore: se quest'ultimo riceve un dato che non soddisfa alla condizione di parità, si tratta di un dato modificato dal canale, e quindi errato.
Per aumentare la sicurezza della trasmissione si possono inserire ulteriori controlli, come ad esempio inviare, dopo un prefissato numero di dati, un dato di controllo ricavato dai precedenti con delle regole prestabilite sia per il trasmettitore che per il ricevitore. In ricezione, se il dato di control-

lo ricevuto e quello ricavato dai dati veri e propri coincidono, la trasmissione è considerata corretta; in caso contrario si richiede una ritrasmissione.
Invece nella trasmissione sincrona, spesso, per il controllo, si inviano alla fine del blocco dati dei caratteri ricavati con un determinato algoritmo dai dati veri e propri che costituiscono quel record: anche in questo caso lo stesso algoritmo è eseguito dal ricevitore ed il carattere che ne risulta deve coincidere con quello ricevuto perché la trasmissione sia considerata corretta. E' ovvio che tutte queste tecniche sono valide in termini probabilistici e cioè assicurano la validità del test con una certa probabilità: potrebbe anche succedere che l'algoritmo adottato confermi l'esattezza della trasmissione quando invece si sono verificati dei particolari errori; si consideri, come esempio, il caso del bit di parità: se all'interno del carattere trasmesso il canale introduce un errore che complementa il valore di due bit, il ricevitore non è in grado di rilevarlo.
Quando è stato illustrato ha il solo scopo di fornire le caratteristiche essenziali dei principali tipi di trasmissioni seriali.
Si tratta ora di vedere come è possibile realizzare una trasmissione seriale mediante un elaboratore il quale deve essere in grado sia di trasmettere che di ricevere.
La trasmissione può anche essere gestita per via software utilizzando come hardware solo un bit di una porta di uscita parallela. Si consideri ad esempio di dover trasmettere un carattere di 8 bit in modo asincrono; le operazioni che l'unità centrale deve svolgere possono essere così schematizzate:

a) prelevare, probabilmente dalla memoria, il byte da trasmettere e calcolarne, se necessario, il bit di parità;

b) inviare un valore logico zero (START) al bit della porta utilizzato come linea di uscita per la trasmissione;

c) attivare un meccanismo di temporizzazione che mantenga il livello sulla linea di uscita per il tempo necessario, quest'ultimo funzione della velocità di trasmissione stabilita;

d) inviare all'uscita il valore del successivo bit del carattere.

Da questo momento in poi il microprocessore deve eseguire i passi da c) a d) tante volte quanti sono i bit che formano il carattere da trasmettere (fig. 8.2), ed infine esso deve inviare il numero stabilito di bit di STOP.
Durante la trasmissione il microprocessore è sempre impegnato dato che la gestione è eseguita via software: si può evitare ciò, e quindi permettere al microprocessore di svolgere altri compiti, utilizzando un timer che, una volta trascorso il tempo di trasmissione di un bit, effettui una richiesta di interruzione per l'invio di un nuovo bit.

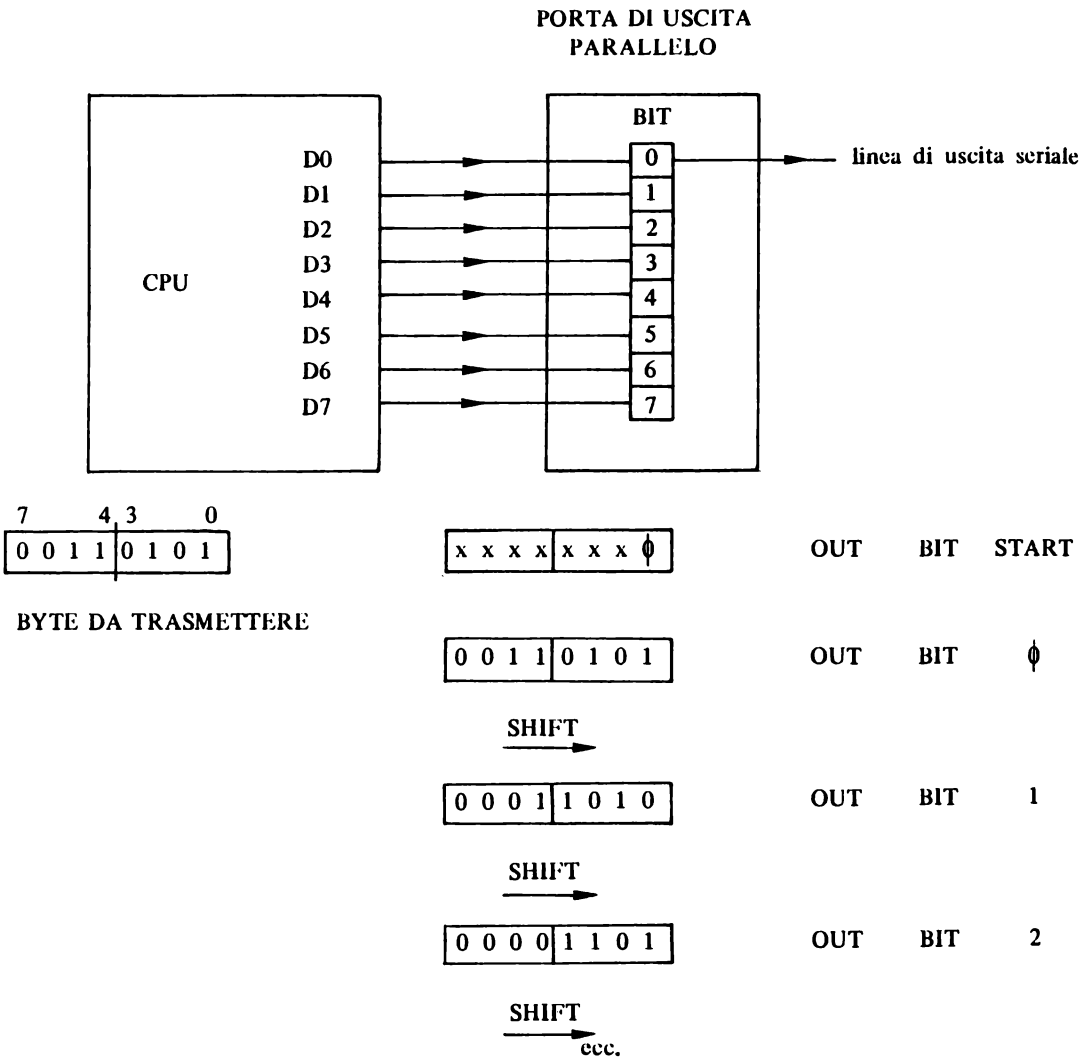



Fig. 8.2
Trasmissione asincrona: le operazioni svolte dalla CPU.

Dato che il tempo di attesa per il servizio di una richiesta di interruzione non è fisso con quest'ultima soluzione si può avere una variazione della durata di ogni singolo bit, con il rischio di far perdere il sincronismo al ricevitore e quindi introdurre degli errori.

Malgrado ciò la soluzione software consente una notevole flessibilità di funzionamento: con semplici modifiche di programma si può variare sia il numero di bit trasmessi per ogni carattere sia il tipo di parità utilizzato, oltre alla velocità di trasmissione. Per quanto riguarda quest'ultima esistono però dei limiti dato che essa diminuisce all'aumentare del numero di prestazioni richieste, come ad esempio il controllo degli errori, ecc.
Per la trasmissione sincrona nel caso si volesse organizzarla via software le complicazioni aumentano sia per l'alto numero di bit che costituiscono un record sia per le strette specifiche temporali proprie di questo tipo di trasmissione.
Per risolvere questi problemi sono stati messi in commercio un certo numero di circuiti integrati che riducono in modo considerevole l'impiego richiesto all'unità centrale per effettuare una trasmissione o ricezione seriale. Tali componenti prendono vari nomi: UART (universal asynchronous receiver/transmitter) che permette di organizzare una trasmissione asincrona; USRT (universal synchronous receiver/transmitter) che gestisce trasmissioni di tipo sincrono; USART (universal synchronous asynchronous receiver/transmitter) mediante il quale è possibile programmare il tipo di trasmissione sincrona o asincrona che si desidera.
Come al solito questi componenti sono realizzati per un determinato processore e la loro utilizzazione con altri tipi di unità centrali richiede in genere qualche adattamento hardware.
Tali dispositivi effettuano in modo autonomo tutte quelle operazioni, necessarie per l'organizzazione di una trasmissione seriale, a partire dai dati forniti in forma parallelo.
Anche in ricezione essi, in modo autonomo, svolgono la conversione da serie in parallelo, oltre a controllare la correttezza della trasmissione.
Per utilizzare tali dispositivi l'unità centrale deve inizialmente predisporre il funzionamento desiderato mediante una opportuna programmazione; durante la trasmissione la CPU deve solo inviare i dati in parallelo al trasmettitore su segnalazione di quest'ultimo; in ricezione preleva solo i dati già predisposti in parallelo dal ricevitore stesso.

8.2. Z80-SIO

Lo Z80-SIO è un dispositivo programmabile, della famiglia Z80, il quale può gestire due canali di comunicazione indipendenti ed è in grado di svolgere la maggior parte delle funzioni necessarie ad un trasferimento di dati in forma seriale, sia in modo sincrono che asincrono, inserendo automaticamente i caratteri di controllo che eventualmente siano richiesti.
Oltre alle linee di ricezione e trasmissione sono presenti altri segnali di controllo che permettono di interfacciare, ad esempio, un floppy disk

con l'aggiunta di una logica esterna molto semplice. E' altresì possibile l'impiego del SIO con un DMA controller per aumentare la velocità di trasferimento di informazioni con la memoria di massa qualora si utilizzino delle periferiche più veloci di un floppy-disk.
Uno schema a blocchi semplificato dello Z80-SIO è riportato nella fig. 8.3; come si vede esiste un blocco di interfaccia il quale attua il collegamento con il bus del sistema e permette sia il trasferimento di dati e comandi, sia quello dei segnali di controllo.

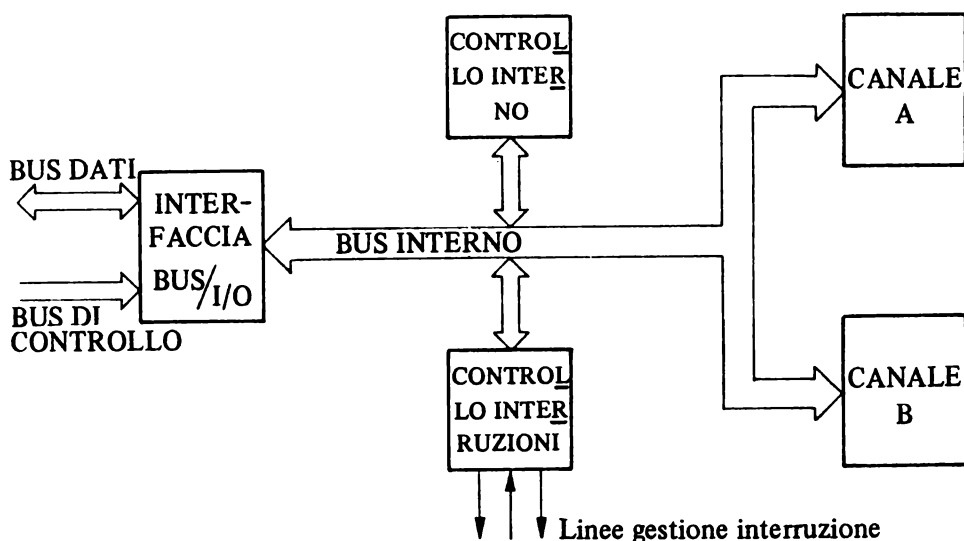



Fig. 8.3
Schema a blocchi semplificato dello Z80-SIO.

Il blocco di controllo interno presiede invece allo scambio di informazioni tra le varie unità funzionali del SIO.
E' presente inoltre un blocco per la gestione delle interruzioni che funziona con le solite modalità proprie dei componenti della famiglia Z80.
Lo schema a blocchi di ogni canale è riportato nella fig. 8.4.
Si può notare una sezione che gestisce la trasmissione ed una che si occupa della ricezione; in entrambe sono presenti un certo numero di registri, di scrittura o di lettura, mediante i quali la CPU rispettivamente programma il tipo di funzionamento desiderato o verifica lo stato della comunicazione in corso.

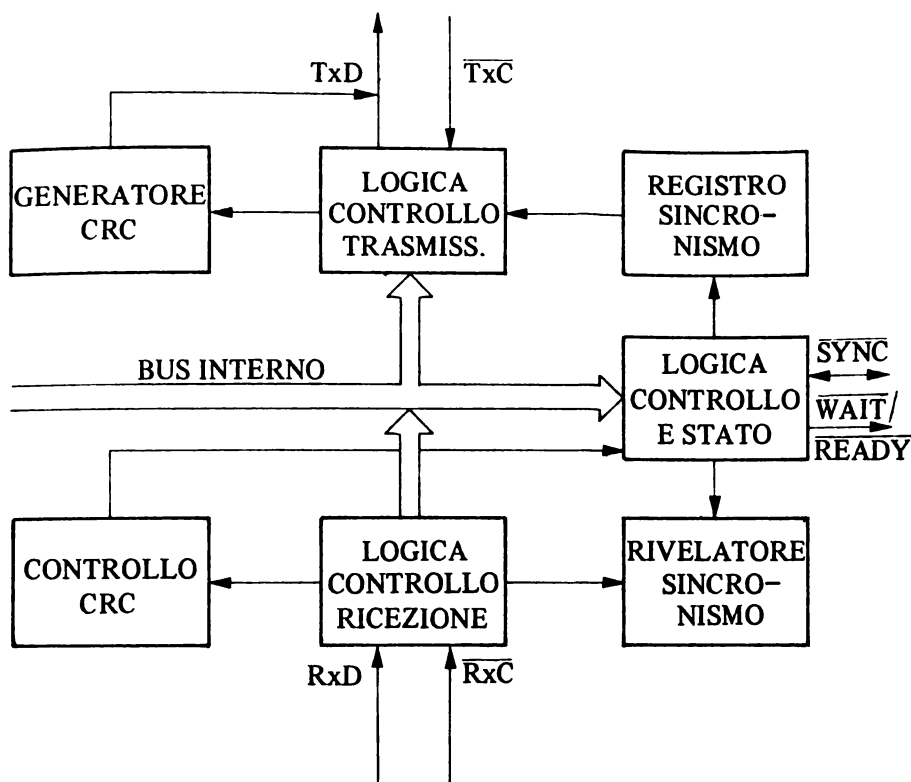



Fig. 8.4
Schema a blocchi di un canale dello Z80-SIO.

I vari segnali associati ai piedini della SIO sono indicati nella fig. 8.5.
Sono presenti, oltre a D0-D7, destinati al collegamento col bus dei dati i segnali IORQ/, M1/ e RD/, generati di solito dalla CPU, che indicano al SIO se l'operazione in corso è di scrittura, di lettura oppure di accettazione di una richiesta di interruzione; le modalità con cui sono utilizzati questi segnali sono uguali a quelle già viste per gli altri componenti della famiglia Z80 come la PIO ed il CTC.
L'ingresso CE/ permette di selezionare il componente ed è attivato con un segnale ottenuto da una decodifica degli indirizzi.
Infine l'ingresso RESET/ consente di imporre al SIO un determinato stato iniziale.
Il segnale presente all'ingresso C/D informa il SIO se quanto è presente sul bus dei dati deve essere interpretato come comando o come dato,

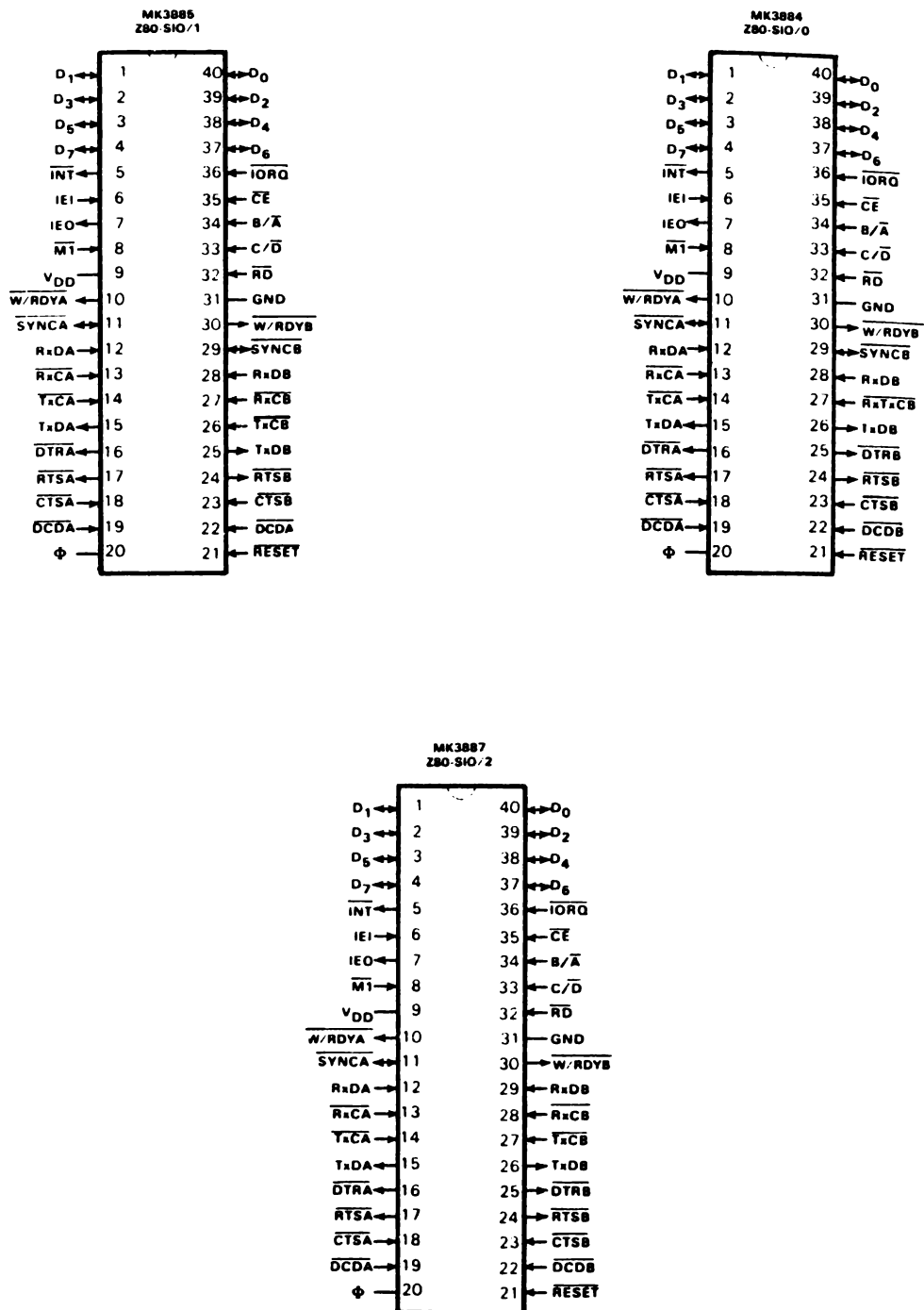



Fig. 8.5
Le diverse versioni dello Z80-SIO. (Mostek, Z80, Microcomputer Data Book 1981).

mentre il segnale sull'ingresso B/A indica a quale canale tale informazione si riferisce. Questi due ingressi sono di solito collegati alle linee A0 e A1 del bus degli indirizzi e permettono di utilizzare perciò quattro indirizzi distinti facilitando in tal modo il software di programmazione del SIO e quello relativo alla gestione della rice-trasmissione.
Per quanto riguarda la comunicazione seriale, in ogni canale è presente una linea per inviare in uscita i dati seriali, (TxD, transmitter data) ed una per riceverli (RxD, receiver data).
Esiste inoltre un ingresso di clock mediante il quale si impone la velocità di trasmissione e di ricezione, che possono essere diverse. Sono a disposizione poi una serie di segnali di controllo, per un eventuale modem, che evitano di dover realizzare una interfaccia ad esso destinata.
E' anche previsto il collegamento con un DMA controller oppure con periferiche particolarmente lente; il segnale W/RDY ha infatti una duplice funzione: come RDY serve per attivare il funzionamento di un DMA, mentre, come WAIT, serve per inserire nel funzionamento della CPU degli stati di attesa. Anche il segnale SYNC può assumere varie funzioni a seconda del modo di funzionamento scelto per il SIO e può essere considerato o di ingresso o di uscita a seconda dei casi.
Dato il numero limitato di piedini (40) non è possibile avere a disposizione tutti i segnali di controllo contemporaneamente e, d'altra parte, non sempre nelle diverse applicazioni servono tutti questi segnali: per questo motivo sono disponibili diverse versioni di SIO, che differiscono fra loro per alcuni dei segnali di controllo.

8.3. Funzionamento dello Z80-SIO

La funzione principale del SIO è quella di convertire dei dati da serie a parallelo e viceversa; nell'eseguire questo compito esso può operare con modalità diverse per soddisfare alle diverse caratteristiche delle comunicazioni di tipo seriale.
La programmazione del SIO, necessaria per stabilire uno dei suoi possibili modi di funzionamento, si attua con l'invio nei registri di controllo di opportune configurazioni di bit.

8.3.1. *Funzionamento asincrono*

I parametri che devono essere inviati al SIO prima che una comunicazione asincrona possa avere inizio sono numerosi: si ricordano qui solo le principali caratteristiche che si possono scegliere. Ogni carattere trasmesso o ricevuto può essere formato da un numero di bit che può variare da 5 a 8: la scelta dipende in genere dal tipo di codifica adottato. Inoltre è previsto l'eventuale inserimento del bit di parità: si può scegliere fra la parità pari oppure dispari.

Il numero di bit di STOP da inserire alla fine di ogni carattere può essere 1, 1,5 o 2 a seconda delle periferiche utilizzate. La frequenza di rice-trasmissione può essere programmata in modo che sia pari alla frequenza del clock utilizzato, oppure sia 1/16, 1/32 o 1/64.
Inviati tutti questi parametri nei registri ad hoc del SIO, assieme a quello che impone il modo di funzionamento asincrono, si deve infine abilitare il SIO stesso all'operazione di trasmissione e di ricezione, e, nel caso della trasmissione, si deve inoltre fornire il dato da trasmettere.
La linea di trasmissione, tenuta nello stato di inattività a livello logico alto, è portata dal SIO a livello basso per l'invio del bit di START.
Esso procede quindi all'invio dei bit del carattere, all'inserimento dell'eventuale bit di parità e dei bit di STOP.
In ricezione onde evitare false interpretazioni di un inizio di una comunicazione, a causa di disturbi presenti nella linea, il SIO considera valido il bit di START se il livello basso si mantiene per un tempo maggiore della metà della durata di un bit con la velocità adottata.
Ogni carattere in una trasmissione asincrona con il SIO può essere costituito come rappresentato in fig. 8.6:

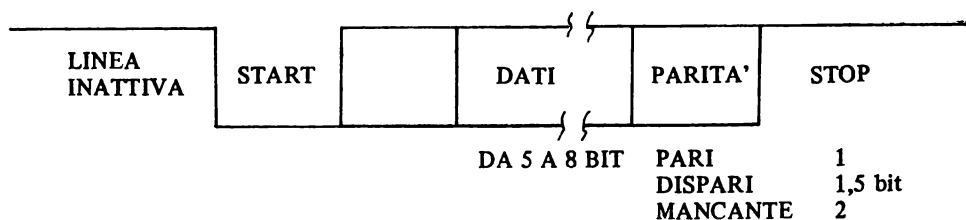



Fig. 8.6
Le possibilità di trasmissione asincrona del SIO.

la successione di caratteri si presenta come in fig. 8.7.

Fig. 8.7
Successione di caratteri trasmessi in modo asincrono.

Come ogni altro componente della famiglia Z80 si può abilitare anche il SIO a fare richieste di interruzione al verificarsi di determinate condizioni: la CPU deve essere predisposta a funzionare nel modo 2 per cui è necessario inviare nel SIO il vettore che serve per l'individuazione della routine di

servizio. Il SIO è in grado di fornire un vettore diverso per ogni causa di interruzione, rendendo così semplice ed agevole la gestione delle richieste relative ai due canali.
In ricezione il SIO può generare interruzioni a seconda di come è stato programmato: una interruzione per ogni dato ricevuto, o solamente all'arrivo del primo dato di un blocco, oppure al verificarsi di condizioni particolari, come ad esempio nel caso di un errore di parità, della perdita di sincronismo, ecc.
I dati ricevuti sono memorizzati su una memoria composta da tre registri che funzionano come una FIFO: ciò al fine di permettere una certa elasticità di intervento nella lettura dei dati da parte della CPU. In ogni caso, se non vengano letti in tempo la ricezione continua con la distruzione però dei dati precedentemente memorizzati e con la segnalazione di una condizione di errore alla CPU. Ovviamente il SIO può solamente segnalare il verificarsi di una determinata condizione di errore, mentre l'eventuale correzione dipende dalla particolare applicazione e deve essere gestita dalla CPU mediante un opportuno programma.

8.3.2. *Funzionamento sincrono*

Il SIO è in grado di gestire vari modi di comunicazioni sincrone: in ogni caso la velocità utilizzata, sia in trasmissione che in ricezione, è uguale a quella del clock: non è cioè possibile ridurre la velocità via software, come nel caso asincrono.
La sincronizzazione avviene premettendo al blocco di caratteri di trasferire uno o due caratteri di sincronismo; per permettere inoltre il controllo sulla correttezza della comunicazione alla fine di un blocco sono aggiunti due caratteri opportuni.
Si hanno quindi due possibili modi di funzionamento, monosync e bisync, come è indicato nella fig. 8.8.

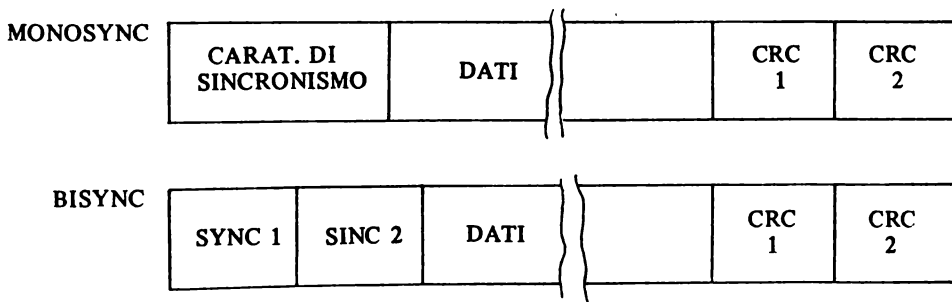



Fig. 8.8
Formato delle trasmissioni monosync e bisync.

Ovviamente i caratteri di sincronismo devono essere precedentemente inviati in due registri interni del SIO, durante la programmazione dello stesso.
Esiste anche una terza possibilità di funzionamento sincrono nella quale non sono richiesti questi caratteri di sincronismo, ma la stessa funzione è svolta da un segnale opportuno presente sul piedino SYNC: in questo caso è compito dell'hardware esterno garantire la sincronizzazione della comunicazione. Negli altri casi invece il ricevitore inizia a immagazzinare i dati solamente dopo che ha ricevuto e riconosciuto i caratteri di sincronismo.
In trasmissione, se il SIO è stato abilitato a richiedere interruzioni, queste sono generate ogni volta che il buffer di trasmissione si svuota, allo scopo di avvisare la CPU di fornire nuovi dati. In alternativa si può anche utilizzare l'uscita WAIT/READY: questa può essere collegata direttamente all'ingresso WAIT della CPU e indica se il SIO è in grado o meno di accettare un nuovo dato di trasmettere. WAIT/READY può anche essere collegata ad un controllore di DMA, nel qual caso indica che il SIO ha il buffer di uscita vuoto ed è pronto a ricevere un nuovo dato trasferito dal controllore di DMA stesso.
Come è stato detto, se il buffer di trasmissione è vuoto, viene inviata una segnalazione alla CPU: si può anche verificare che quest'ultima non sia in grado di fornire il dato richiesto in tempo utile: il SIO, in questo caso, per non far perdere il sincronismo al ricevitore, invia dei byte o di sincronismo o di controllo. Il vantaggio di questo modo di procedere è una maggiore flessibilità nella trasmissione, anche se richiede una adeguata organizzazione software per la gestione dei dati in ricezione.

8.3.3. *Ricezione sincrona a caratteri*

Una volta abilitato il SIO alla ricezione esso si mette in stato di ricerca o di attesa: in questa situazione continua a controllare la linea di ricezione e acquisisce effettivamente i dati solo dopo aver riconosciuto i caratteri di sincronismo inviati dal trasmettitore. Anche questi caratteri devono essere stati memorizzati all'interno del SIO in fase di programmazione.
Per il trasferimento dei dati ricevuti alla CPU si possono seguire diverse modalità: oltre alle richieste di interruzione, si può anche fare in modo che all'arrivo di un nuovo dato sia settato un bit specifico dei registri di stato, demandando alla CPU il compito di andare a leggere questo registro per procedere o meno alla lettura del dato ricevuto.
Nel caso di trasferimento di un blocco di dati si può programmare il SIO a segnalare l'arrivo solamente del primo dato del blocco; questo modo di procedere trova una conveniente applicazione quanto la velocità di trasferimento è particolarmente elevata per cui una volta iniziata la ricezione i dati devono essere prelevati in modo continuo dalla CPU. Si può inoltre utilizzare l'uscita WAIT collegata alla CPU, in modo che, attivata la rou-

tine di ricezione, la CPU stessa rimane a disposizione del trasferimento dei dati per tutta la durata della comunicazione.

8.3.4. *Funzionamento sincrono SDLC o HDLC*

Il modo di funzionamento sincrono appena descritto permette il trasferimento di caratteri e prende perciò il nome di "character-oriented". Attualmente si sta imponendo un'altra tecnica di trasmissione sincrona che permette il trasferimento di record composti da un numero qualsiasi di bit. Questo tipo di comunicazione prende il nome di "bit-oriented" e presenta vari vantaggi, in particolare nella trasmissione dati fra elaboratori. Esistono diversi protocolli per questo tipo di comunicazione ed il SIO è in grado di gestirne due in particolare; lo SDLC (Synchronous Data Link Control) e l'HDLC (High Level Synchronous Data Link Control).
I dati trasmessi con queste due tecniche non solo devono essere preceduti e seguiti da alcune informazioni per il controllo e per la gestione della comunicazione, ma subiscono anche delle modifiche in fase di trasmissione, modifiche che il ricevitore deve automaticamente eliminare per fornire il dato originario.
L'inserimento dei controlli e le modifiche accennate è attuato automaticamente dal SIO, una volta programmato allo scopo, così che la CPU deve solo fornire i dati da trasmettere o acquisire quelle ricevuti. Il messaggio da trasmettere è spezzato in record, o pacchetti, che non devono superare una lunghezza fissata.
Il formato di un record è riportato nella fig. 8.9.

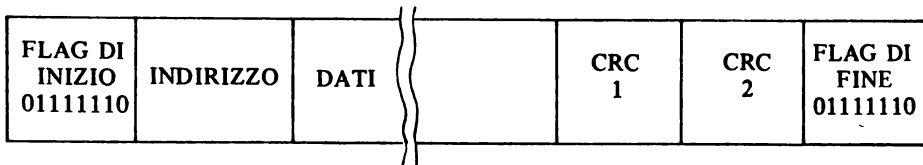



Fig. 8.9
Formato di un record in una trasmissione sincrona del tipo "bit-oriented".

Il primo carattere trasmesso è un byte che prende il nome di flag, il quale presenta la caratteristica di avere i 6 bit centrali al valore logico 1: questa configurazione di 6 bit consecutivi al valore 1 è ammessa solo per il flag; segue un byte di servizio dove in genere è contenuto l'indirizzo di destinazione del pacchetto.
Sono poi trasmessi i bit dei dati in sequenza: la caratteristica fondamentale che essi devono presentare è che non possono essere presenti più di 5 bit di valore 1 consecutivi. Se nel trasferimento di certi dati si trova una tale combinazione di 1 il SIO spezza la sequenza inserendo un bit 0 dopo il quinto bit al valore 1.

Seguono quindi due byte di controllo ed il pacchetto termina con un byte di flag, uguale a quello iniziale, che potrebbe anche costituire il flag di inizio del pacchetto successivo.
Prima di iniziare una trasmissione è ovviamente necessario programmare il SIO per specificare i vari parametri che caratterizzano un pacchetto, come ad esempio il numero di bit che compongono il campo dei dati, ecc.
La CPU deve inviare al SIO, oltre ai dati, anche l'indirizzo, mentre l'organizzazione del pacchetto, come l'inserimento degli eventuali zeri ed i byte finali per il controllo e di chiusura, è gestita autonomamente senza impegnare la CPU.

8.3.5. *Ricezione SDLC e HDLC*

Anche in questo caso è naturalmente necessaria una precedente programmazione in modo da predisporre il SIO a funzionare con determinate caratteristiche.
Il SIO analizza lo stato della linea di ricezione, per verificare quando è presente un byte di flag: se ciò accade il SIO acquisisce il successivo byte e lo confronta con l'indirizzo precedentemente programmato da parte della CPU; per l'accettazione dei dati successivi e quindi per l'attivazione della ricezione è necessaria la coincidenza fra questi due indirizzi. Se in certe applicazioni non è sufficiente un indirizzo di 8 bit, questo deve essere spezzato in due parti: i primi 8 bit vanno gestiti come ora detto, mentre gli altri sono considerati dati da parte del SIO, per cui il successivo controllo deve essere effettuato dalla CPU; il SIO non ha cioè la possibilità di variare la lunghezza del campo indirizzi.
I dati man mano ricevuti sono controllati dal SIO, che elimina gli zeri eventuali inseriti in trasmissione.
Alla CPU è quindi presentato un record privo degli elementi di controllo: sono infatti presenti solamente il byte di indirizzo ed i dati, con lo stesso formato con cui sono stati inviati al SIO trasmittente. Alla fine del record ricevuto il SIO controlla i due byte CRC ed eventualmente segnala le situazioni di errore.
Il trasferimento dei dati dal SIO alla CPU può avvenire con le modalità già viste nel caso di trasmissione sincrona a caratteri; se non sono state abilitate le richieste di interruzione, la CPU può controllare un bit di stato del SIO, che indica se un dato è stato ricevuto. Se invece le richieste di interruzione sono state attivate si può avere una interruzione o al primo byte oppure per ogni byte ricevuto.
Si ripete in fig. 8.10 uno schema più dettagliato di un canale del SIO, dove sono evidenti i blocchi funzionali che presiedono a quanto finora descritto.
Per la programmazione e le caratteristiche di questo componente si rimanda ai manuali messi a disposizione dai costruttori.

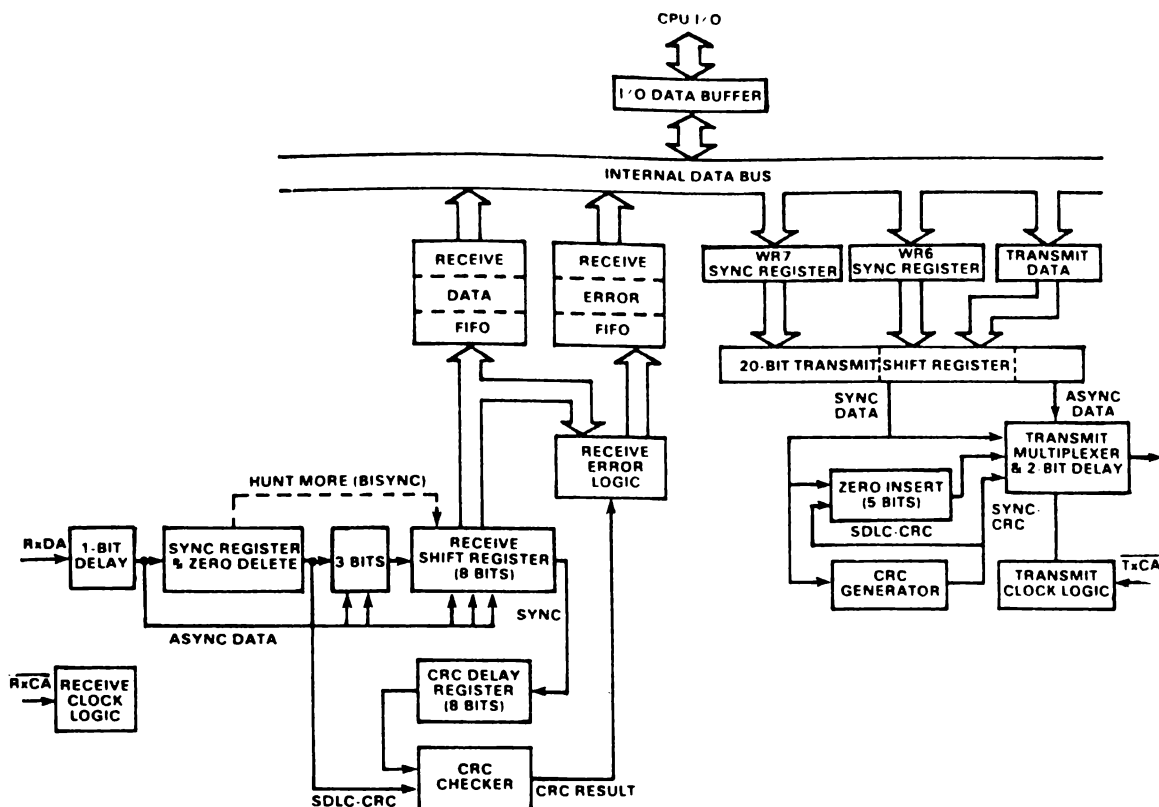



Fig. 8.10
Schema a blocchi di un canale del SIO.
(Mostek, Z80, Microcomputer Data Book, 1981).

Come si è visto il SIO è in grado di soddisfare notevoli esigenze e può adattarsi alla maggior parte dei modi di trasmissione seriale attualmente in uso. Si ricorda qui che la trasmissione HDLC soddisfa alla quasi totalità delle norme X25 per la trasmissione usata nelle reti a commutazione di pacchetto. Con l'impiego del SIO e di un adeguato software è quindi possibile anche il collegamento a questo tipo di reti.
In pratica non sempre in un unico componente si richiedono e si utilizzano tutte le prestazioni fornibili dal SIO, in quanto difficilmente si avrà la necessità di passare da una trasmissione asincrona ad una sincrona: sono stati perciò messi in commercio dei componenti che presentano solo alcune caratteristiche del SIO; ad esempio lo Z80 SIO/9 ha le caratteristiche

di un SIO, ma con un solo canale di comunicazione, mentre lo Z80 DART è un SIO in grado di funzione solamente in modo asincrono.

8.4. Altri dispositivi per la ricetrasmissione seriale

Per i microprocessori della serie INTEL è disponibile un USART il cui schema a blocchi è riportato nella fig. 8.11.

D7 D0	Data Bus (8 bits)
C/D	Control or Data is to be Written or Read
RD	Read Data Command
WR	Write Data or Control Command
CS	Chip Enable
CLK	Clock Pulse (TTL)
RESET	Reset
TxC	Transmitter Clock
TxD	Transmitter Data
RxC	Receiver Clock
RxD	Receiver Data
RxRDY	Receiver Ready (has character for 8080)
TxRDY	Transmitter Ready (ready for char from 8080)

DSR	Data Set Ready
DTR	Data Terminal Ready
SYNDET/BD	Sync Detect/ Break Detect
RTS	Request to Send Data
CTS	Clear to Send Data
TxE	Transmitter Empty
Vcc	+5 Volt Supply
GND	Ground

Fig. 8.11
Schema a blocchi dell'8251. (Intel, Component Data. Catalog 1979).

E' possibile il funzionamento asincrono e sincrono ma in quest'ultimo caso solamente del tipo 'character-oriented'; è inoltre presente un solo canale con clock esterni indipendenti per la trasmissione e per la ricezione.
E' inoltre prevista la possibilità di utilizzare un modem e sono resi disponibili i vari segnali che servono in questo caso.
I segnali di controllo con la CPU, anche se concettualmente sono simili a quelli visti per la SIO, sono adatti ai microprocessori per i quali questo

componente è stato realizzato. Nel modo di funzionamento asincrono si ritrovano le stesse caratteristiche già esaminate nel caso del SIO: è possibile trasmettere o ricevere caratteri composti da 5, 6, 7 oppure 8 bit; si può inserire o meno il bit di parità come pure si può variare il numero di bit di STOP inseriti alla fine della trasmissione di ogni carattere. Tutte queste opzioni sono stabilite dalla CPU nella fase di inizializzazione mediante invio di adatti comandi di programmazione.

Nella trasmissione e ricezione sincrona è possibile stabilire il sincronismo con dei segnali hardware esterni, oltre che con l'invio di uno, o due, caratteri di sincronismo. Questo componente non calcola i caratteri di controllo finali, per cui è demandata al software la verifica del corretto procedere della comunicazione.

Per i processori della Motorola sono invece stati realizzati dei componenti che sono in grado di assolvere ad una sola delle funzioni che sono riscontrate nella SIO. Ad esempio l'MC6850 è in grado solo di gestire trasmissioni seriali asincrone. Le modalità di funzionamento sono pressoché identiche a quelle già viste per gli altri componenti, tenendo presente che la comunicazione con la CPU è stata organizzata per i microprocessori della serie Motorola. Sono previste diverse velocità di trasmissione e l'uso di modem, oltre alla segnalazione delle diverse condizioni di errore che si possono verificare.

Invece l'MC6852 permette solamente il trasferimento di dati seriali in modo sincrono a carattere. Anche in questo caso il sincronismo può essere ottenuto sia da una struttura esterna oppure mediante l'inserzione di uno o due caratteri di sincronismo. L'MC6852 possiede all'interno tre registri organizzati a FIFO sia in trasmissione che in ricezione e questa caratteristica è molto utile nel caso si utilizzino le richieste di interruzione perché concedono un maggior tempo per l'intervento da parte della CPU.

Infine l'MC6584 permette di gestire la comunicazione sincrona sia SDLC che HDLC, in quanto inserisce automaticamente i flags iniziali e finali, gli zeri nei dati se sono presenti sequenze con più di 5 bit consecutivi, ecc.

Per una dettagliata descrizione di tutti questi componenti si rimanda ai manuali messi a disposizione dal costruttore.

Capitolo 9

L'ACCESSO DIRETTO ALLA MEMORIA

9.1. Il trasferimento di dati in un microelaboratore

Nella struttura di un microelaboratore si è finora sempre considerato presente solo un master, la CPU, con il compito di coordinare le attività dei vari dispositivi slave, cioè le memorie e le porte di I/O, programmabili o meno, eventualmente presenti.
E' tuttavia possibile progettare un microelaboratore in modo che siano presenti più di un dispositivo che possa assumere la funzione di master; i motivi che consigliano tale tipo di organizzazione sono parecchi: non sempre la CPU, pur con la sua notevole flessibilità di operare, è in grado di svolgere certe particolari funzioni con quella velocità e grado di efficienza che possono essere richiesti.
Un tipico esempio è il trasferimento di dati, ad alta velocità, tra la memoria ed una periferica, per il tramite di una porta di I/O.
In questo caso è preferibile ricorrere all'utilizzazione della tecnica del DMA (Direct Memory Access), la quale, anche se richiede una struttura hardware esterna relativamente complicata, consente di ottenere delle prestazioni notevoli per quanto riguarda la velocità di trasferimento di dati con periferiche.
Si supponga, ad esempio, che si debbano trasferire un certo numero di dati dalla memoria ad una porta di uscita; per questa operazione se non si usa la tecnica del DMA è necessario organizzare un adatto programma, il cui diagramma di flusso potrebbe essere quello riportato nella fig. 9.1.
Una volta depositati in opportuni registri i valori dei parametri che identificano sia il numero di byte, o dati, da trasferire, sia la locazione di memoria da cui inizia l'area interessata al trasferimento, sia la porta di I/O, la CPU inizia un ciclo di trasferimento di un dato: per far ciò essa deve effettuare un certo numero di letture della memoria.
La prima azione che la CPU effettua è quella di leggere il dato dalla memoria e depositarlo in un suo registro interno; per far ciò essa deve prelevare il codice operativo della relativa istruzione, e ciò può richiedere da uno a tre accessi alla memoria di programma, a seconda del tipo di istru-

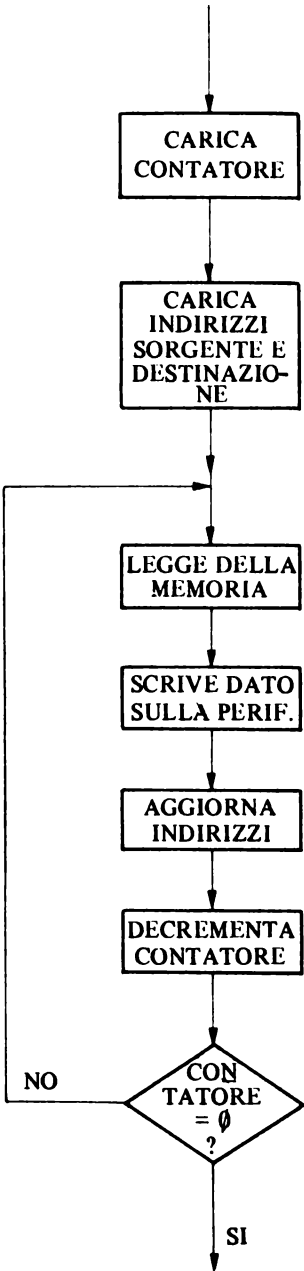



Fig. 9.1
Diagramma di flusso per il programma di trasferimento di un blocco di dati da memoria a porta di uscita.

zione. A questi si deve aggiungere un ulteriore accesso, questa volta alla memoria dei dati, per prelevare effettivamente il dato: in conclusione per depositare in un suo registro interno un dato l'unità centrale deve effettuare un numero di accessi alla memoria che può variare da un minimo di due ad un massimo di quattro. A questo punto la CPU può trasferire il dato testé prelevato alla porta di uscita e per far ciò sono ancora necessari uno o più accessi alla memoria, a seconda del tipo di istruzione utilizzato.
Poiche si vuole trasferire un blocco di dati, devono essere aggiornati i diversi parametri necessari per il trasferimento del dato successivo: deve essere quindi aggiornato sia l'indirizzo della locazione di memoria da cui prelevare il dato sia il contatore del numero di byte da trasferire. La CPU deve inoltre eseguire un controllo per verificare se è stato completato lo spostamento dell'intero blocco.
In conclusione il trasferimento di un dato, che da un punto di vista logico richiede solo un ciclo macchina di lettura da memoria ed uno di scrittura sulla porta di uscita interessata, avviene invece mediante l'esecuzione di molti ciclo macchina.
Ciò comporta una notevole riduzione della velocità del trasferimento rispetto al caso ideale, e ciò può portare a delle limitazioni nella utilizzazione di periferiche particolarmente veloci quali sono appunto i dischi magnetici, i terminali video, ecc.
I difetti propri di tale modo di procedere sono in parte superati se si usano dei microprocessori i quali presentino delle istruzioni, particolarmente potenti, che consentono il trasferimento di blocchi di dati. Ad esempio col microprocessore Z80 si può utilizzare l'istruzione OTIR la quale, una volta che sia stata eseguita una preliminare inizializzazione dei registri interni, necessari, al solito, per indicare la sorgente da dove prelevare il dato e il numero di dati da trasferire, fa si che automaticamente la CPU prelevi il dato, lo invii alla porta di uscita, aggiorni l'indirizzo di memoria e il contatore di dati trasferiti e, se quest'ultimo non ha raggiunto lo zero, attui il trasferiemnto di un nuovo dato. Si noti che in questo caso una sola istruzione realizza tutte le operazioni indicate nel diagramma di fig. 9.1: ciò fa si che la velocità di trasferimento dei dati sia notevolmente aumentata, in quanto sono evitati gli accessi in memoria necessari per il prelevamento delle singole istruzioni.
Da quanto detto potrebbe sembrare non necessario utilizzare una tecnica di trasferimento più veloce, come il DMA, in particolare se si considera che la velocità di trasferimento ottenibile con queste istruzioni è paragonabile a quella che si raggiunge con certi tipi di controllori di DMA.
Si consideri ora invece il caso in cui il trasferimento di dati avviene tra un disco magnetico e la memoria centrale: l'unità a disco, una volta che sia correttamente posizionata la testina di lettura, richiede alla CPU di iniziare il trasferimento mediante una richiesta di interruzione. Tra l'istante in cui

è attivata tale richiesta e quello in cui la CPU è in grado di acquisire un dato può trascorrere un intervallo di tempo, dell'ordine della decina di microsecondi, il quale non può essere determinato a priori: infatti il tempo di accettazione della richiesta risulta variabile al variare del ciclo macchina eseguito dalla CPU nell'istante in cui la richiesta di interruzione è stata attivata. Ciò non è accettabile per il disco, in quanto esso è in grado di fornire i dati, con frequenza costante, entro un certo tempo massimo dall'istante in cui esso ha dato avviso alla CPU di essere pronto al trasferimento. Anche in questo caso sono possibili diverse soluzioni, sempre utilizzando un solo master, cioè la CPU.
Ad esempio si potrebbe usare la tecnica del polling: l'unità centrale continua a testare un bit di stato della unità a dischi e quando quest'ultima segnala la disponibilità di un dato la CPU dà inizio al relativo trasferimento. Si potrebbero utilizzare le istruzioni:

```
LOOP:   IN      STATO
        JR      Z, LOOP
```

In questo caso il ritardo massimo è di due istruzioni che, complessivamente, necessitano per la loro esecuzione un numero di periodi di clock inferiore a quello previsto come caso peggiore utilizzando la tecnica dell'interruzione; un altro vantaggio di questa tecnica è che non è necessario provvedere al salvataggio dello stato della macchina. Il grosso inconveniente di questa soluzione è dovuto al fatto che il microprocessore è completamente impegnato a testare lo stato della periferica e non può eseguire altri compiti.
Sono disponibili però dei dispositivi in grado di eseguire trasferimenti di dati in modo veloce e con tempi di attesa minimi, senza richiedere una particolare struttura hardware aggiuntiva: sono questi i DMAC (Direct Memory Access Controller); essi, una volta inizializzati, cioè programmati da parte della CPU, quando entrano in azione assumono il controllo dei bus del sistema e quindi assumono la funzione di master.

9.2. Trasferimento in DMA

Le tecniche di trasferimento utilizzate nei vari DMAC sono sostanzialmente tre: "sequenziale", "simultanea", "trasparente".

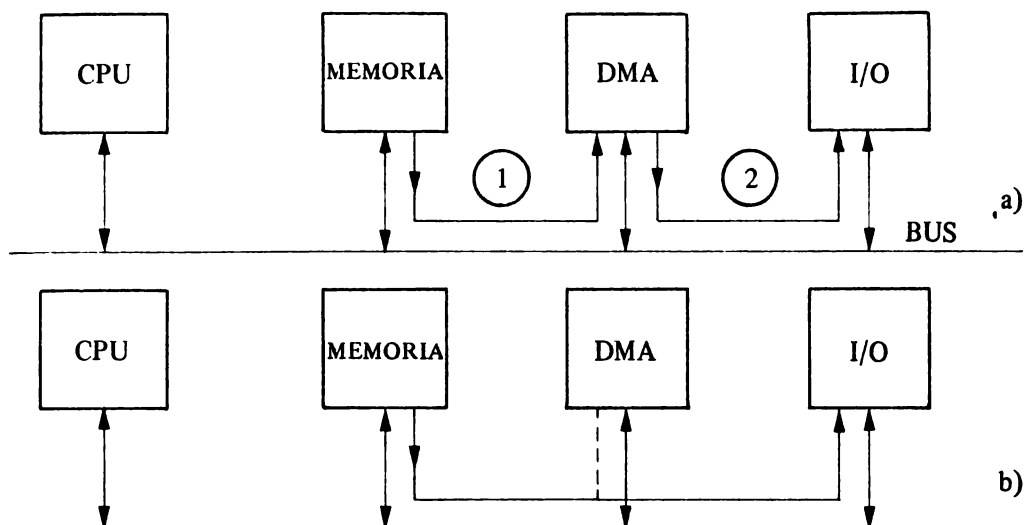



Fig. 9.2
a) trasferimento sequenziale b) trasferimento simultaneo

In fig. 9.2a è rappresentato il trasferimento sequenziale di un dato dalla memoria verso una periferica; il DMAC, assunto il controllo dei bus, esegue una operazione di lettura da memoria all'indirizzo precedentemente specificato, memorizzando il dato ottenuto su un suo registro interno. Successivamente esegue una operazione di scrittura su una porta di uscita inviando così il dato alla periferica interessata.
Con la tecnica "simultanea", o flyby, invece il DMAC effettua la lettura della memoria: quest'ultima mette a disposizione il dato richiesto sul bus dei dati, come in un normale ciclo di lettura. A questo punto il DMAC informa la periferica di prelevare il dato in quel momento presente sul bus dei dati, eliminando in questo modo sia l'operazione di memorizzazione in un suo registro interno sia la successiva operazione di scrittura.
E' una tecnica sicuramente più veloce rispetto al caso precedente ma richiede una struttura hardware più complessa.
In entrambe le tecniche appena descritte la unità centrale è bloccata nel suo funzionamento dato che il DMAC ha il totale controllo dei bus del sistema.

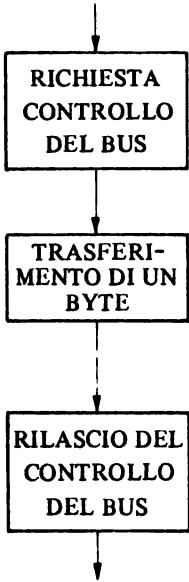



Fig. 9.3
Trasferimento di un singolo dato (a byte).

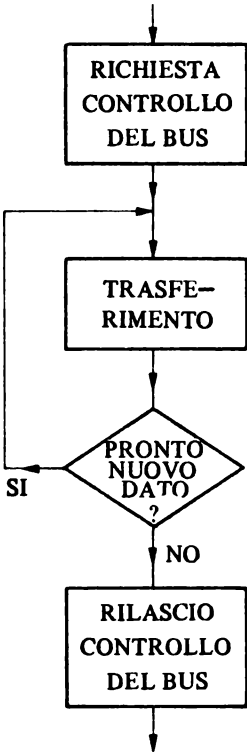



Fig. 9.4
Trasferimento di un blocco di dati (a burst).

Per quanto riguarda il trasferimento "trasparente" si deve ricordare che nel funzionamento di una CPU esistono degli intervalli di tempo pari a qualche periodo di clock se non addirittura a degli interi cicli macchina, durante i quali il bus non è occupato per il trasferimento di informazioni: è allora possibile sfruttare per il trasferimento di dati da memoria a periferiche e viceversa tali intervalli di tempo.
Tale tecnica richiede ovviamente una opportuna organizzazione hardware che non tutti i tipi di microprocessori possono permettere; rispetto alle due tecniche precedenti non solo si ha una diminuzione della velocità di trasferimento dei dati, ma in certi casi tale velocità non è costante, dipendendo essa dal tipo di istruzioni in esecuzione; per questi motivi può trovare applicazione solo quando non sono richieste particolari caratteristiche di sincronismo al trasferimento.
Nelle tre tecniche descritte si possono adottare vari modi di funzionamento: a byte, a pacchetti di byte (burst), e continuo.
Nel primo modo una volta ottenuto il controllo dei bus si procede al trasferimento di un solo byte, dopo di che i bus sono rilasciati. Ovviamente il DMAC aggiorna gli indirizzi ed i contatori, per cui al successivo trasferimento è tutto predisposto per una corretta esecuzione.
Tale modo di funzionamento è indicato nel diagramma di flusso di fig. 9.3.
Negli altri due modi di funzionamento è invece richiesto un segnale dalla periferica che indichi quando la stessa è pronta per il trasferimento di un dato.
Il diagramma di flusso per il modo a burst è indicato nella fig. 9.4.
Una volta preso il controllo del bus il DMAC esegue un trasferimento e quindi verifica se è possibile procedere al successivo: se ciò è vero lo esegue ed effettua poi una nuova verifica; se la risposta è negativa rilascia il bus.
Il funzionamento in modo continuo è invece rappresentato nella fig. 9.5.
In questo caso il bus viene rilasciato solamente una volta ultimato il trasferimento del blocco di dati precedentemente programmato: eseguito il trasferimento di un dato il DMAC conserva il controllo del bus anche se non è pronto il dato successivo, e si mette in attesa fino a quando questo è reso disponibile.
E' ovvio che la velocità di trasferimento complessiva aumenta dal primo al terzo modo di procedere; contemporaneamente però si ha una corrispondente aumento del tempo in cui l'unità centrale è bloccata per cui i vari modi di funzionamento vanno scelti in base alle particolari specifiche richieste.
Il limite principale nell'uso della tecnica del DMA consiste proprio nel fatto che l'unità centrale rimane bloccata: non è quindi possibile servire alcuna richiesta di interruzione, in quanto durante il rilascio dei bus il microprocessore non analizza nessuno dei segnali di controllo esterni, ad ec-

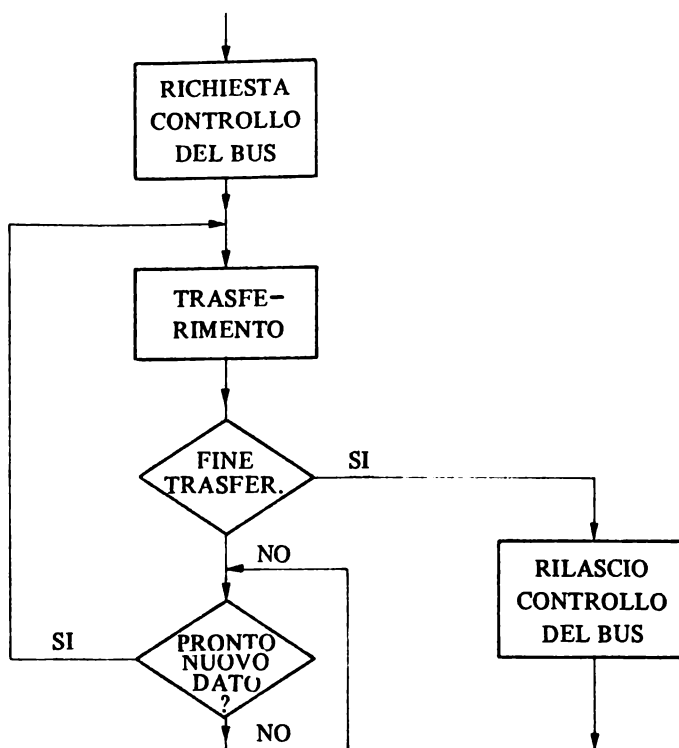



Fig. 9.5
Trasferimento di dati in modo continuo.

cezione di quello che lo informa di quando i bus non sono più utilizzati. Non è inoltre possibile procedere al rinfresco delle memorie dinamiche, non solo nel caso in cui è il processore che controlla direttamente tale attività, ma anche quando è stato predisposto un adatto circuito esterno, se non ricorrendo a dell'hardware opportuno che permetta di sincronizzare tale operazione con il funzionamento del DMAC.
Non sempre ciò è accettabile e gli inconvenienti che si verificano dipendono in particolare dal modo di funzionamento scelto.
Nel "byte mode" infatti si ha la massima riduzione di tali inconvenienti in quanto si ha un funzionamento alternato fra CPU e DMAC a livello di ciclo macchina o intervallo di tempo ad esso paragonabile.
Nel "burst mode" invece il tempo in cui la CPU è bloccata dipende dalle caratteristiche della periferica e precisamente dal fatto che questa, una volta eseguito un trasferimento, sia in grado o meno di effettuarne immediatamente un successivo. Si potrebbero verificare anche dei tempi di inattività per la CPU molto lunghi in particolare nel caso si debbano trasferire dei blocchi di dati di dimensioni considerevoli.
Si può ovviare a questo inconveniente intervenendo sul segnale di dato

pronto condizionandolo con altre segnalazioni, come ad esempio richieste di interruzioni da parte di altre periferiche. E' ovvia la complicazione hardware che ne può risultare.
In conclusione un vantaggio del "burst mode" è che esso tiene impegnato il bus solamente quando effettivamente serve per il trasferimento.
Se invece si utilizza il modo continuo non è possibile nessun intervento da parte di altri dispositivi e quindi esso, anche se consente la massima velocità di trasferimento, deve essere impiegato quando ne sono anche accettabili gli inconvenienti.
Non va dimenticato inoltre che i DMAC devono essere programmati: nel caso di applicazioni dove si richiede un frequente aggiornamento dei parametri deve essere tenuto presente anche il tempo necessario per tale operazione se si vuole valutare correttamente le prestazioni che si possono ottenere.

9.2.1. *Segnali di controllo in un DMAC*

I segnali che servono per la gestione di un generico DMAC sono indicati nella fig. 9.6, dove sono stati messi in risalto alcuni comandi specifici di questo componente.

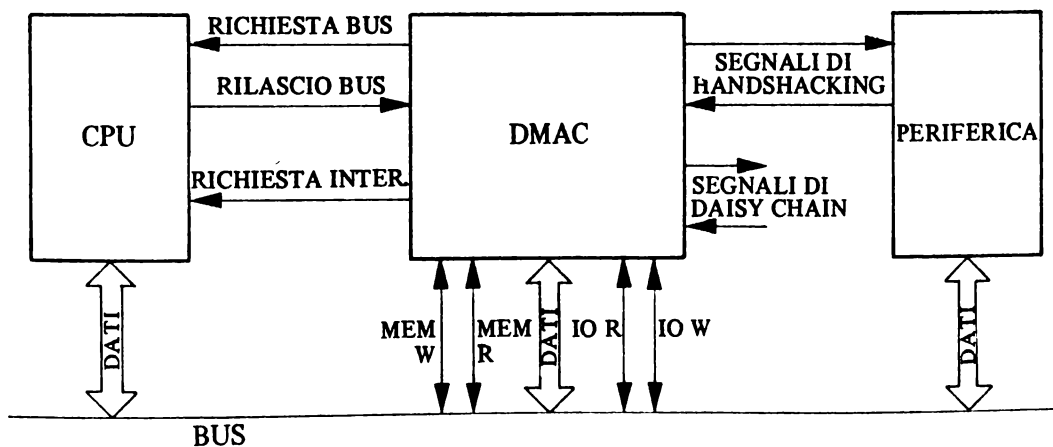



Fig. 9.6
I segnali di un DMAC.

Nei confronti dell'unità centrale il DMAC deve essere in grado di richiedere il rilascio del controllo del bus e di ricevere la conferma dell'avvenuto rilascio.
Nei confronti delle periferiche invece, oltre ai vari segnali di lettura e scrittura, presenti sulle linee utilizzate per programmare e selezionare il DMAC stesso, sono spesso presenti dei segnali di handshaking che permettono una più semplice gestione del trasferimento.
Dato che in un elaboratore possono essere presenti più dispositivi in grado di assumere la funzione di master occorre, in questo caso, risolvere eventuali situazioni di richieste multiple: si possono nello scopo adottare vari criteri uno dei quali è una gestione in daisy-chain.
Anche la individuazione della sorgente e destinazione del trasferimento può avvenire con tecniche diverse. Una soluzione semplice consiste nel procedere alla selezione attraverso lo stesso bus degli indirizzi con modalità simili a quelle adottate dalla CPU. Con questa soluzione qualsiasi periferica presente nel sistema può trasferire dei dati tramite il DMAC. In altri casi invece la periferica è individuata solamente tramite segnalazioni particolari, che ovviamente possono consentire di sfruttare il DMAC solamente per un numero limitato di dispositivi; si noti che quest'ultima soluzione rende più semplice il trasferimento con la tecnica del "flyby". Come al solito anche le richieste di interruzione possono essere gestite in modo diverso a seconda del tipo di DMAC utilizzato.

9.3. Ciclo di richiesta e rilascio del bus

Si vogliono ora analizzare le modalità per la richiesta ed il rilascio del controllo del bus prendendo in considerazione il comportamento del microprocessore Z80. In fig. 9.7 è riportato il relativo diagramma temporale dove sono indicati i vari segnali che interessano in questa situazione.
Il processore controlla lo stato della linea BUSRQ/ sul fronte di salita dell'ultimo periodo di clock di ogni ciclo macchina; se tale segnale è attivo la CPU non manda in esecuzione il successivo ciclo macchina: si interrompe, cioè, l'esecuzione dell'istruzione in corso. Le uscite relative alle linee di indirizzo e di dato, e quelle che generano i segnali di controllo sono poste nello stato di alta impedenza. Successivamente la CPU segnala all'esterno l'assunzione di questo nuovo stato rendendo attivo il segnale BUSAK/.
Tutte le attività della CPU sono ora bloccate ad eccezione di quella che controlla, sul fronte di salita di ogni periodo di clock, lo stato della linea BUSRQ/: finché questa rimane attiva la CPU permane nello stato di inattività. Se invece BUSRQ/ assume un livello alto, la CPU disattiva il segnale BUSAK/ e riprende l'esecuzione dell'istruzione sul fronte di salita del successivo periodo di clock. Dall'attivazione della richiesta BUSRQ/ al rilascio del controllo del bus, intercorre un intervallo di attesa pari alla durata di

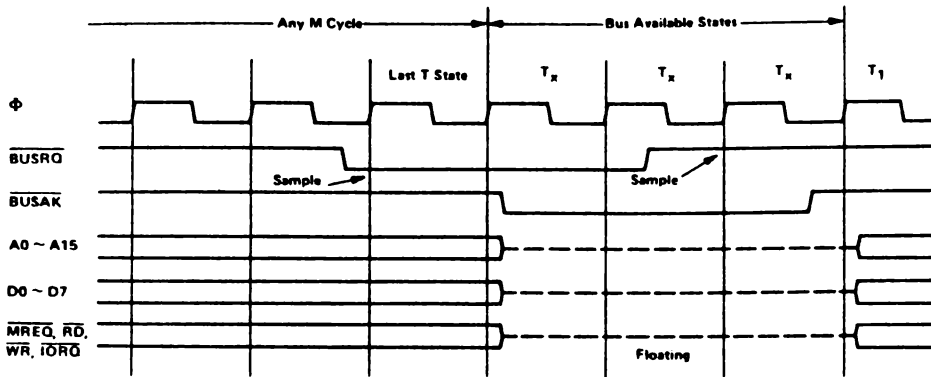



Fig. 9.7
Ciclo di richiesta e rilascio del bus nel microprocessore Z80 (Mostek, Microcomputer Data Book, 1981).

un ciclo macchina, cioè al massimo 6 periodi di clock, mentre dalla disattivazione di BUSRQ/ alla ripresa delle attività della CPU si ha, al massimo, una attesa di 2 periodi di clock. Lo stesso comportamento, anche se con differenze nei legami temporali dei vari segnali, si riscontra con gli altri microprocessori.
Come si può notare, lo Z80 può essere tolto dallo stato di inattività solamente con la disattivazione del segnale per la richiesta del rilascio del bus, BUSRQ/: non sono prese in considerazione quindi richieste di interruzione, nè è possibile procedere al rinfresco delle eventuali memorie dinamiche presenti.
Nel caso in cui la richiesta del rilascio del bus avvenga nell'ultimo ciclo macchina di una istruzione e siano contemporaneamente presenti anche delle richieste di interruzione, di qualunque tipo, lo Z80 considera a priorità più elevata la richiesta dei bus; il ritardo massimo di un ciclo macchina, per avere il rilascio del bus, si verifica quindi in tutte le situazioni nelle quali il microprocessore può venire a trovarsi.

9.4. Lo Z80 DMAC

Si vogliono ora illustrare le caratteristiche generali di un DMAC realizzato per essere impiegato in particolare con il processore Z80: lo Z80-DMA;

esso soddisfa alle specifiche dinamiche e utilizza il bus di controllo con le stesse modalità del processore Z80.

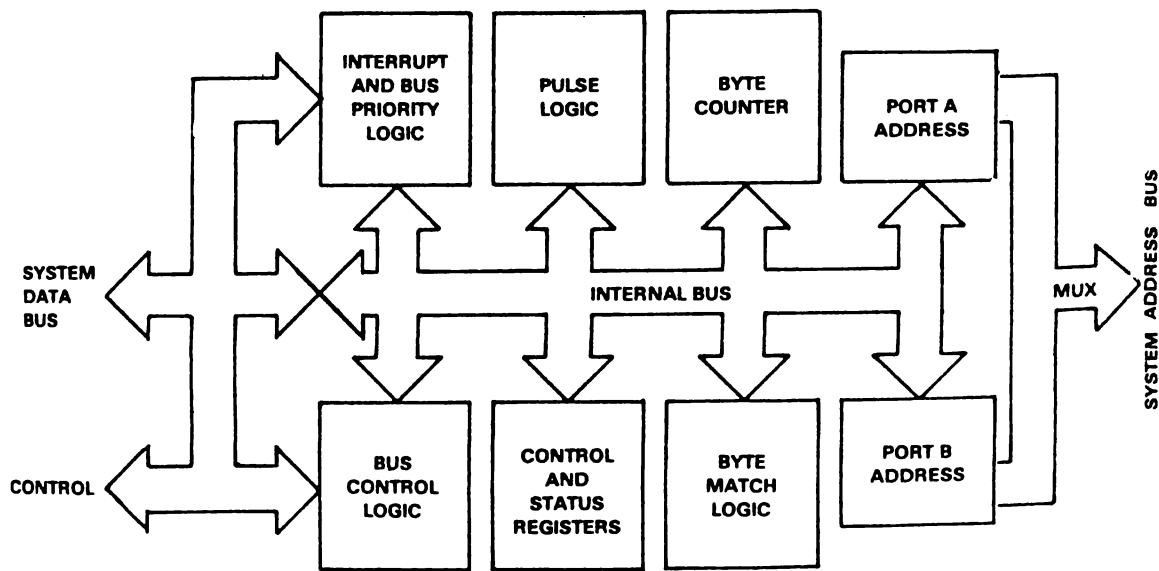



Fig. 9.8
Schema a blocchi dello Z80-DMA (Mostek, Z80 Microcomputer Data Book, 1981).

Lo Z80-DMA, il cui schema a blocchi è riportato in fig. 9.8, invia sul bus degli indirizzi quelli necessari alla selezione o di una cella di memoria oppure di una porta di I/O, interessate al trasferimento di dati, tramite due porte, denominate A e B, le quali possono fornire ciascuna l'indirizzo della sorgente o quello della destinazione.
I dati che devono essere trasferiti transitano ovviamente sul bus dei dati.
Esiste inoltre un "byte counter" (contatore di byte) che contiene in ogni istante il numero di byte ancora da trasferire. Quando tale contatore raggiunge lo zero il DMAC fornisce una segnalazione all'esterno, che conferma l'avvenuto trasferimento del prefissato numero di dati; questa informazione è gestita da un blocco detto "pulse logic".
Esistono poi un blocco che gestisce l'assunzione e il rilascio del controllo del bus, ed uno, per la gestione delle interruzioni, perfettamente analogo a quelli già visti in componenti della stessa famiglia. Un ulteriore blocco di controllo generale ha nel suo interno un insieme di registri utilizzati sia per

la programmazione dello Z80-DMA che per rilevarne lo stato di funzionamento.
Nella fig. 9.9 dono riportati i vari segnali associati ai piedini di questo componente e raggruppati per funzioni.

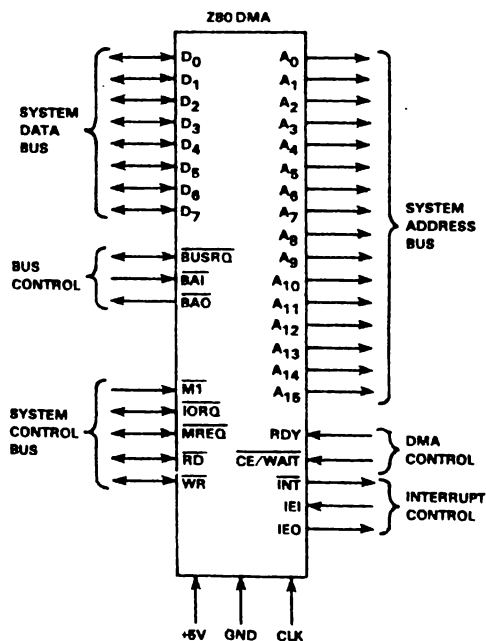



Fig. 9.9
Segnali nello Z80-DMA (Mostek, Z80 Microcomputer Data Book, 1981).

Attraverso i piedini D0-D7, che devono essere collegati al bus dei dati del sistema, transitano due tipi di informazioni: al primo appartengono sia quelle ché servono alla programmazione dello Z80-DMA e sono generate dalla CPU, sia quelle provenienti dal DMAC e che danno conto alla CPU dello stato dell'operazione in corso; il secondo tipo di informazione è costituito dai dati provenienti dalla sorgente e inoltrati alla destinazione da parte del DMAC.
I piedini A0-A15 forniscono i segnali di indirizzamento per individuare i dispositivi sorgente e destinazione collegati al bus ed interessati al trasferimento.
I segnali associati ai piedini M1/, IORQ/, MREQ/, RD/, e WR/, hanno la stessa funzione degli omonimi segnali presenti e nella CPU e nelle periferiche programmabili; si noti però che gli ultimi quattro sono bidirezionali.

Ciò è necessario in quanto il DMAC durante il suo funzionamento da master si sostituisce completamente alla CPU nel controllo del BUS del sistema e quindi deve essere in grado di generare gli opportuni segnali per le operazioni di lettura e scrittura degli slave.
I piedini INT/ IEI/ ed IEO/ servono per la gestione delle interruzioni e per la risoluzione delle loro priorità ed hanno lo stesso significato e funzionamento già analizzati per i componenti della famiglia Z80. Inoltre il piedino INT/ può anche assumere un significato diverso, durante il funzionamento del DMAC come master, dato che in questa situazione una eventuale richiesta di interruzione non può essere presa in considerazione da parte della CPU. Tramite il piedino INT/ il DMAC può fornire un impulso ogni volta che sono stati trasferiti un certo numero programmato di dati. In tal modo è possibile far conoscere all'esterno il numero di dati già trasferiti, senza dover effettuare una lettura del contatore interno del DMAC, cosa del resto impossibile se il DMAC sta funzionando come master, per cui la CPU non può in alcun modo leggere lo stato di tale contatore.
I segnali BAI/, BAO/, e BUSRQ/ di ingresso, di uscita e bidirezionale, rispettivamente, sono utilizzati per effettuare la richiesta del rilascio del bus e sono gestiti ed analizzati da tutti i controllori di DMA eventualmente presenti nel sistema in modo tale che solo uno di essi può ottenere il controllo dei bus.
Il segnale CE/ WAIT assume due diversi significati a seconda dello stato in cui si trova il DMAC: nel caso che esso non abbia il controllo del bus tale segnale ha l'usuale significato di selezione del componente interessato allo scambio di informazioni con la CPU, sotto il controllo della stessa.
In questo stato infatti la CPU vede il DMAC come una normale porta di ingresso e di uscita e l'attivazione di CE/ può essere ottenuta con uno dei metodi, già visti, di decodifica degli indirizzi e dei segnali di controllo.
Nel caso invece il controllore DMA abbia assunto la funzione di master un segnale presente nell'ingresso WAIT serve per adeguare la velocità di trasferimento dei dati alle caratteristiche dei dispositivi che a tale trasferimento sono interessati; analogamente a quanto visto per lo Z80, questo segnale può essere reso attivo dai dispositivi di memoria o di I/O per inserire dei periodi di attesa durante le operazioni di lettura e di scrittura, permettendo così anche ai dispositivi più lenti di inviare o ricevere con successo i dati da trasferire.
Per quanto riguarda il segnale RDY, di ingresso, esso serve per informare il DMAC che il dispositivo interessato è pronto per il trasferimento di dati: le modalità di utilizzazione di tale segnale dipendono dal modo con cui è stato programmato il DMAC.
Finora si sono descritti i segnali associati ai vari piedini presenti nel DMAC; per quanto riguarda la programmazione del dispositivo, esso possiede al suo interno 21 registri, di sola scrittura, che servono per ricevere le informazioni provenienti dalla CPU, ed 8 registri, di sola lettura, che permettono alla CPU di controllare lo stato e del componente e dell'operazione di trasferimenti in corso di esecuzione.

I registri di scrittura sono organizzati in sette gruppi, ognuno contenente un certo numero di registri, come si può vedere alla fig. 9.10.

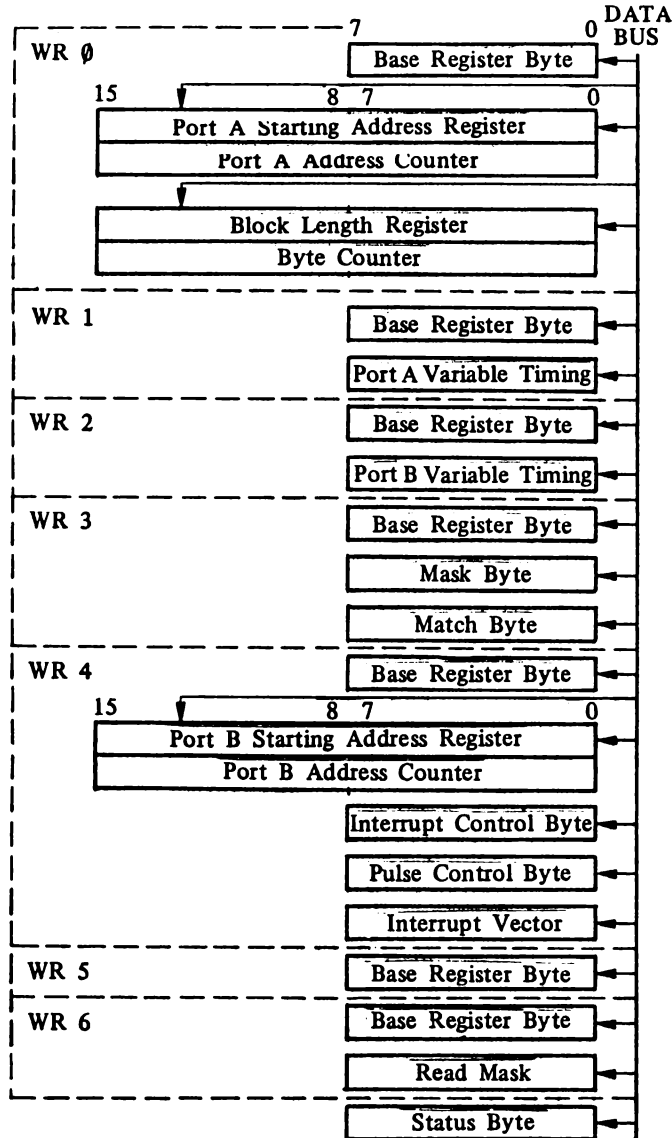



Fig. 9.10
I registri di scrittura nello Z80-DMA (ZILOG, Z80-DMA Technical Manual).

Ogni singolo gruppo ha un registro base, che individua univocamente il gruppo, ed altri registri, in numero variabile da 0 a 6 a seconda del gruppo e cioè delle informazioni che lo caratterizzano.

Per la scrittura e l'indirizzamento di questi registri si deve procedere in due fasi distinte:

– nella prima fase occorre individuare il registro base del gruppo: ciò è ottenuto ponendo ad un valore prefissato alcuni bit del dato che la CPU invia al DMAC. Il significato e la posizione di questi bit è stabilito dal costruttore e si può dedurre da quanto è riportato nella fig. 9.11.

Ad esempio il registro base del gruppo 3 è individuato dal fatto che il dato inviato dalla CPU deve avere D7 = 1 e D1 = D0 = 0, mentre per individuare quello del gruppo 5 occorre un dato in cui sono prefissati ben 5 bit. I rimanenti bit del dato inviato ad un registro base assumono significati diversi, a seconda del registro base cui si riferiscono; in particolare alcuni bit, se posti al valore 1, indicano quali altri registri del gruppo riceveranno successive informazioni da parte della CPU. Se ad esempio il bit D4 del registro base 3 è posto a 1 il successivo dato inviato dalla CPU al DMAC deve essere da questi interpretato come un dato, che deve essere memorizzato nel registro MATCH BYTE, appartenente al gruppo 3, che conterrà la configurazione di bit da cercare quando al DMAC si richiede la funzione di ricerca di un certo dato.
In conclusione, per l'invio dei dati necessari alla programmazione del DMAC la CPU deve procedere con una scrittura iniziale su un registro base e successivamente con l'invio di dati che saranno ordinatamente memorizzati dal DMAC proprio nei registri indicati dal contenuto del registro base. Questo modo di procedere, anche se appesantisce l'organizzazione del software, ha il vantaggio di non richiedere nessun ingresso aggiuntivo per la individuazione dei diversi registri.
Per quanto riguarda l'acquisizione da parte della CPU dello stato del DMAC, esistono 8 registri di sola lettura; per leggere il loro contenuto la CPU deve procedere ancora in due fasi: la prima consiste nello scrivere nel registro base di scrittura del gruppo 6 quali sono i registri interessati alla lettura. Successivamente, ed in modo ordinato, la CPU può procedere alla lettura: ad ognuna di esse il DMAC fornirà il contenuto dei registri, nell'ordine stabilito, saltando ovviamente quelli non richiesti.
Per una dettagliata descrizione dei modi di programmazione del DMAC si rimanda ai manuali forniti dai costruttori; di seguito si illustrano solamente le caratteristiche di funzionamento di questo componente e le prestazioni che da esso si possono ottenere.

9.4.1. *Modi di operare dello Z80 - DMA*

Le operazioni che possono essere effettuate dallo Z80 - DMA si possono

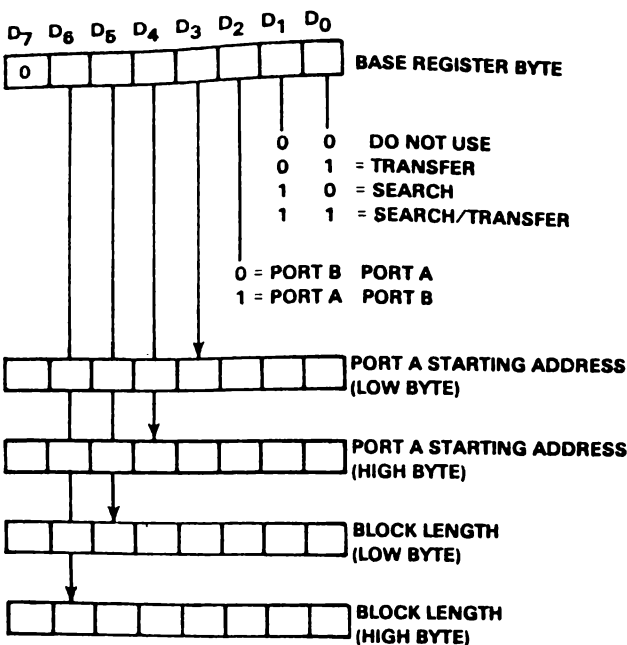
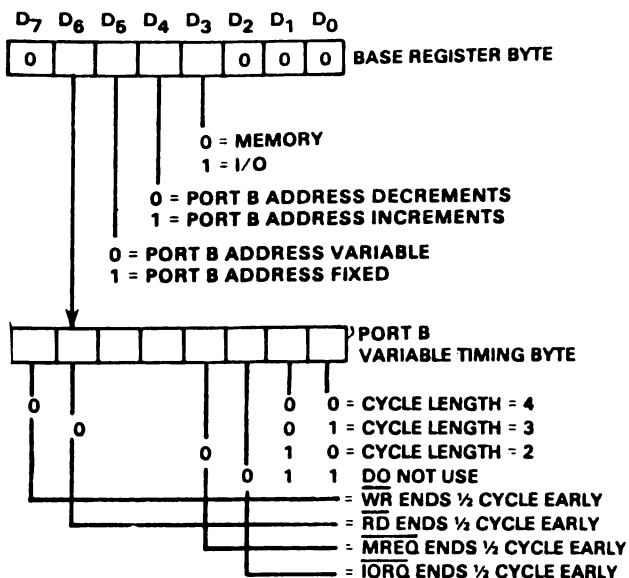
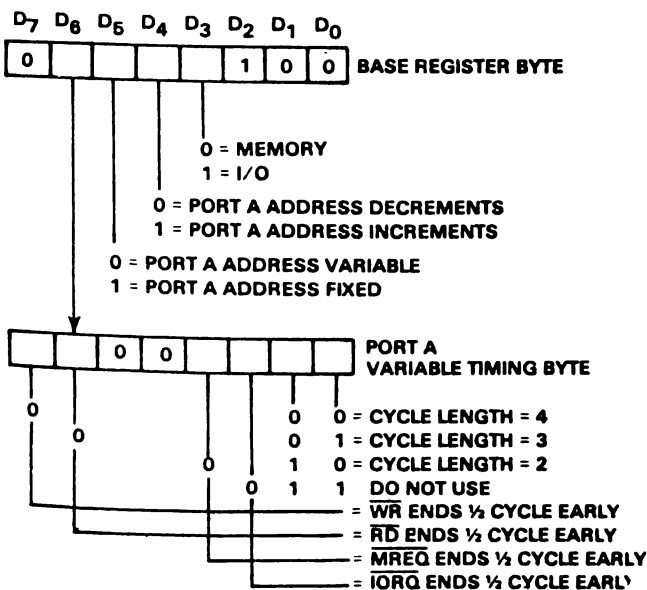
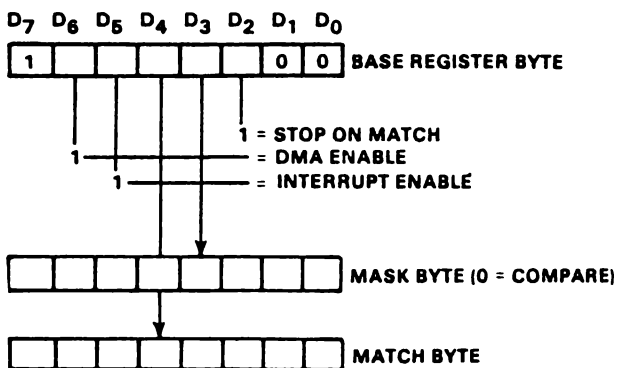



Fig. 9.11a
Programmazione dei registri WR0 ÷ WR3 nello Z80-DMA (Mostek, Z80 Microcomputer Data Book, 1981).

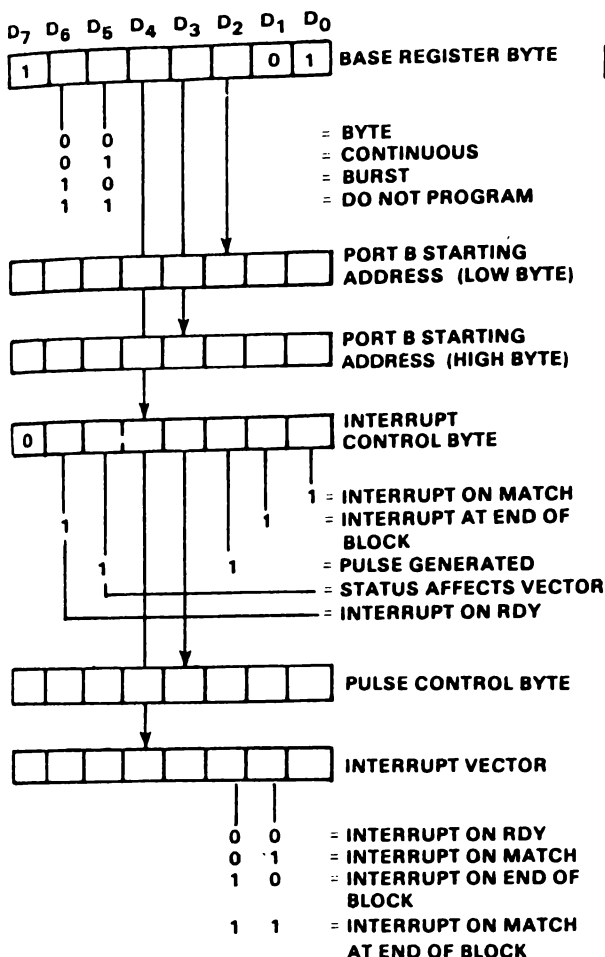
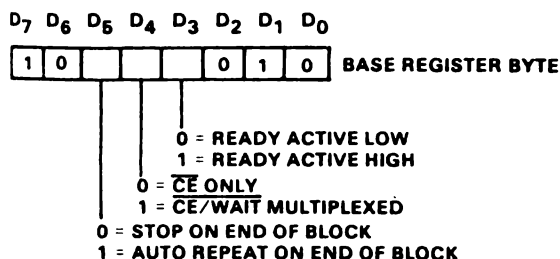
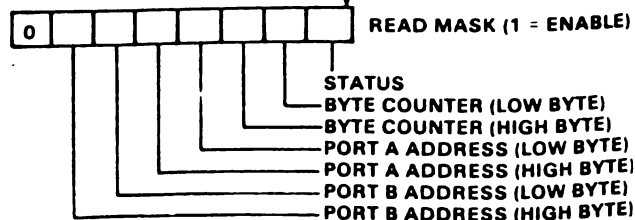
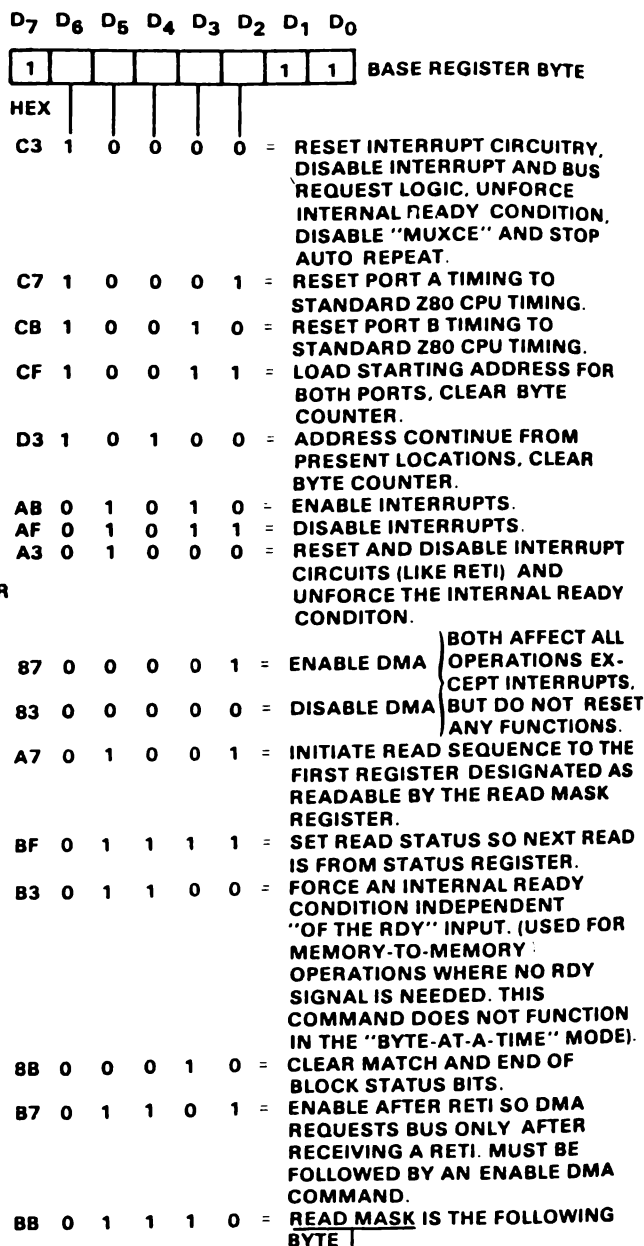



Fig. 9.11b
Programmazione dei registri WR4 ÷ WR6 nello Z80-DMA (Mostek, Z80 Microcomputer Data Book, 1981).

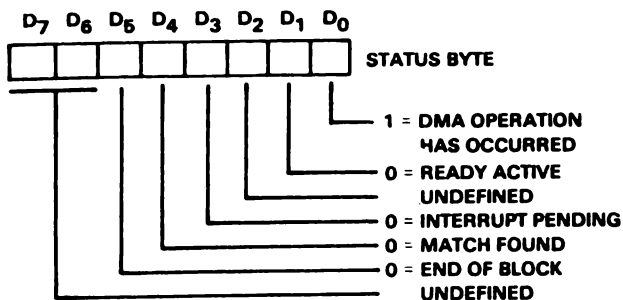
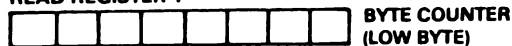
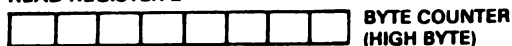
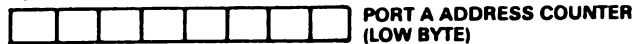
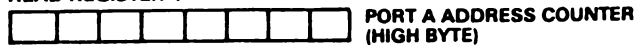
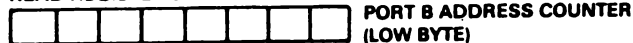



Fig. 9.11c
Significato dei registri di lettura nello Z80-DMA (Mostek, Z80 Microcomputer Data Book, 1981).

suddividere in tre classi:

- trasferimento di dati fra due dispositivi; questi possono essere indifferentemente di memoria o di ingresso-uscita.
- ricerca di un particolare byte in memoria o in una porta di ingresso; tutti i dati letti dalla sorgente sono confrontati, byte per byte, con il contenuto di un registro, interno al DMAC, il quale contiene il byte da ricercare. Il contenuto del registro può essere mascherato di modo che il paragone interessa solo il valore dei bit non mascherati
- trasferimento di dati fra due dispositivi e contemporanea ricerca di un byte.

In fig. 9.12 sono schematizzate tutte le possibili operazioni che lo Z80-DMA può effettuare;

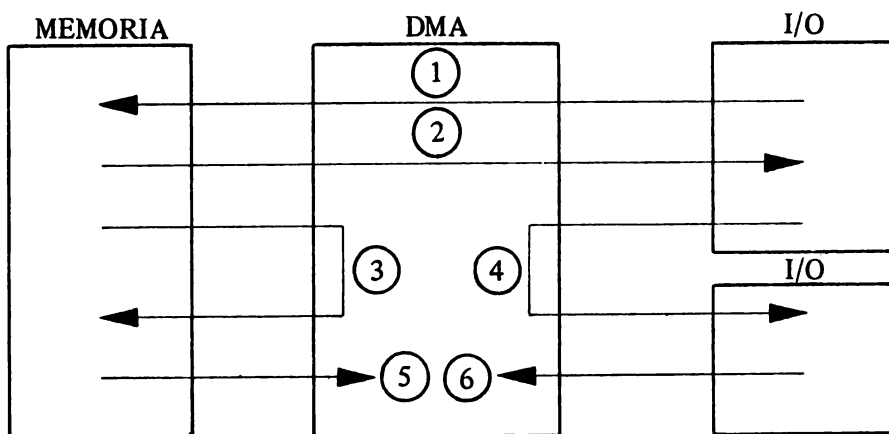



Fig. 9.12
Tipi di trasferimento che possono essere gestiti dallo Z80 - DMA.

Quelli indicati con 1) e 2) rappresentano il trasferimento di dati, da memoria e periferica, o viceversa, con la possibilità di effettuare anche una ricerca di un particolare byte durante il trasferimento. I 3) e 4) indicano invece il trasferimento di dati da memoria a memoria oppure da periferica a periferica rispettivamente. In ogni caso è sempre possibile aggiungere come opzione anche la ricerca di una particolare configurazione di bit in un byte.
Infine i tipi di funzionamento indicati con 5 e 6 consistono solamente in una operazione di ricerca di un byte rispettivamente in memoria oppure su un dispositivo di ingresso.

I dati sono letti dalla sorgente e scritti sulla destinazione un byte alla volta durante una qualsiasi operazione di trasferimento; durante una operazione di sola ricerca il byte è letto e confrontato con quanto contenuto in un registro interno, precedentemente programmato.
L'utilizzazione più corrente di un DMAC è per il trasferimento di dati da memoria ad una periferica o viceversa; il trasferimento da memoria a memoria può essere utilizzato per il trasferimento veloce del contenuto di aree di memoria da una area ad un'altra, ma, molto più spesso, essa è utilizzata per il trasferimento di dati con dispositivi periferici quando si è adottato per questi ultimi la tecnica di decodifica memory-mapped I/O.
Il trasferimento da I/O ad I/O può essere utile qualora sia necessaria una acquisizione di dati molto veloce con contemporaneo salvataggio dei dati acquisiti in una memoria di massa.
Per quanto riguarda la ricerca di una particolare configurazione di bit, questa è utilizzata ad esempio nel trasferimento di dati da disco a memoria. In tale operazione è infatti necessario individuare l'inizio di un settore o di una traccia prima di procedere alla lettura. La segnalazione dell'avvenuto ritrovamento del dato cercato o la fine del trasferimento possono essere segnalati alla CPU con diverse opzioni come sarà illustrato più avanti.
Per il trasferimento generico di un byte sono necessari due indirizzi, quello che individua la sorgente e quello della destinazione; ambedue sono forniti dalla CPU nella fase di inizializzazione. Nei trasferimenti successivi occorre aggiornare gli indirizzi e tale aggiornamento può avvenire in modi diversi: si può prescrivere un incremento, oppure un decremento, automatico degli indirizzi di partenza, oppure mantenere gli indirizzi fissi.
Ad esempio in un trasferimento di dati da memoria a porta di uscita l'indirizzo sorgente deve essere incrementato (o decrementato) ogni volta mentre quello relativo alla destinazione deve rimanere fisso. Comportamento opposto invece si ha nel trasferimento da una porta di ingresso alla memoria.
Nel caso di un trasferimento da memoria a memoria sia l'indirizzo della sorgente che quello della destinazione devono essere aggiornati dopo ogni trasferimento; se i dispositivi interessati sono invece ambedue di I/O i due indirizzi devono rimanere costanti. Queste diverse possibilita sono consentite nello Z80-DMA e, come al solito, sono scelte durante la fase di programmazione.
In fig. 9.12 è descritto il diagramma di flusso di una generica sessione di trasferimento di un byte. La figura vuole solo rappresentare le azioni fondamentali che sono o possono essere intraprese dal DMAC una volta che ha ricevuto il controllo del BUS e prima di rilasciare tale controllo.
Si noti l'aggiornamento degli indirizzi, se richiesto, con possibilità di incremento oppure decremento, e la funzione di ricerca di un dato con segnalazione, mediante un flag, del risultato della ricerca.

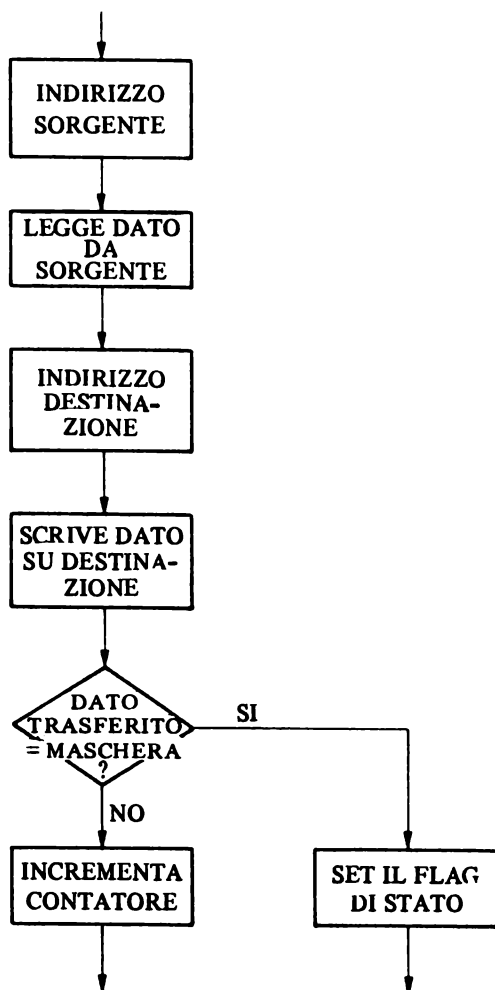



Fig. 9.13
Diagramma di flusso relativo ad una generica operazione di trasferimento di un byte.

In ognuna delle diverse classi di funzionamento ora descritte si possono avere vari modi di operare da parte del DMAC. Questi modi possono essere così catalogati:

9.4.2. *"Un byte alla volta"*

In questo caso il DMAC prende il controllo del bus, trasferisce il dato dalla sorgente alla destinazione, e poi rilascia il controllo del bus.
Per trasferire un successivo byte il DMAC deve effettuare una ulteriore richiesta dei bus: una volta ottenutone il controllo procede come appena detto.

Questo modo di operare, indicato spesso anche come trasferimento singolo, può attivarsi con varie opzioni: ad esempio il DMAC può essere attivato a richiedere il controllo o via software oppure mediante una segnalazione esterna.
Nella fig. 9.14 è riportato un diagramma di flusso delle principali operazioni svolte e delle decisioni prese dal DMAC in questo modo di funzionamento.

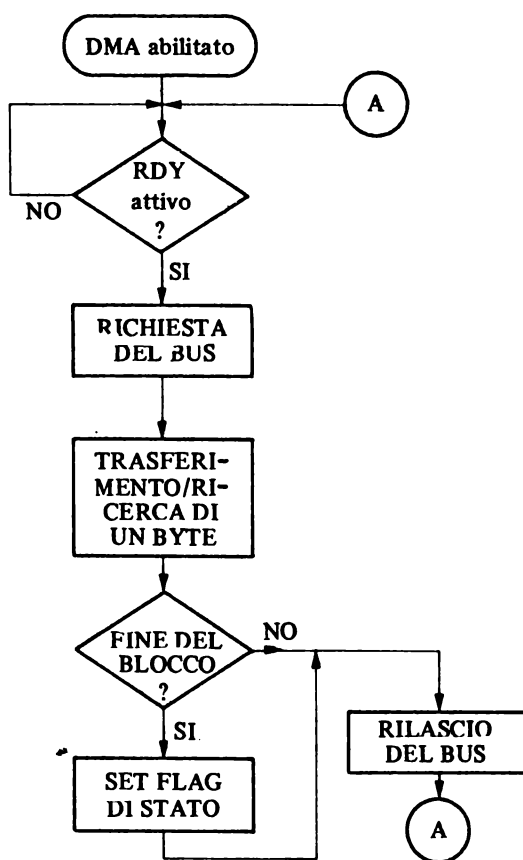



Fig. 9.14
Funzionamento dello Z80-DMA nel trasferimento di un byte alla volta.

Il rilascio del bus dopo ogni trasferimento permette alla CPU di eseguire almeno un ciclo macchina prima di rilasciare nuovamente il controllo del bus se ne viene fatta un'ulteriore richiesta. In questo modo di operare

sembra che la CPU ed il DMA funzionino "contemporaneamente" anche se in realtà ad una analisi più dettagliata questo non è esatto. La velocità di trasferimento è ovviamente rallentata, ma contemporaneamente si permette alla CPU di continuare ad operare e ciò può risultare molto importante nel caso, come già detto, che sia necessario servire richieste di interruzione oppure procedere al rinfresco delle eventuali memorie dinamiche impiegate nel sistema.

9.4.3. *A "pacchetti di byte"*

Un altro modo di operare dello Z80-DMAC è in "burst mode": in questo caso si deve trasferire un blocco di dati e non un solo byte come nel modo precedente. Una volta ottenuto il controllo del bus il DMAC inizia il trasferimento dei dati e lo continua fino a quando rimane attivo il segnale READY, cioè in pratica finché ci sono dei dati pronti per essere trasferiti. Un diagramma di flusso di questo modo di operare è indicato nella fig.9.15.

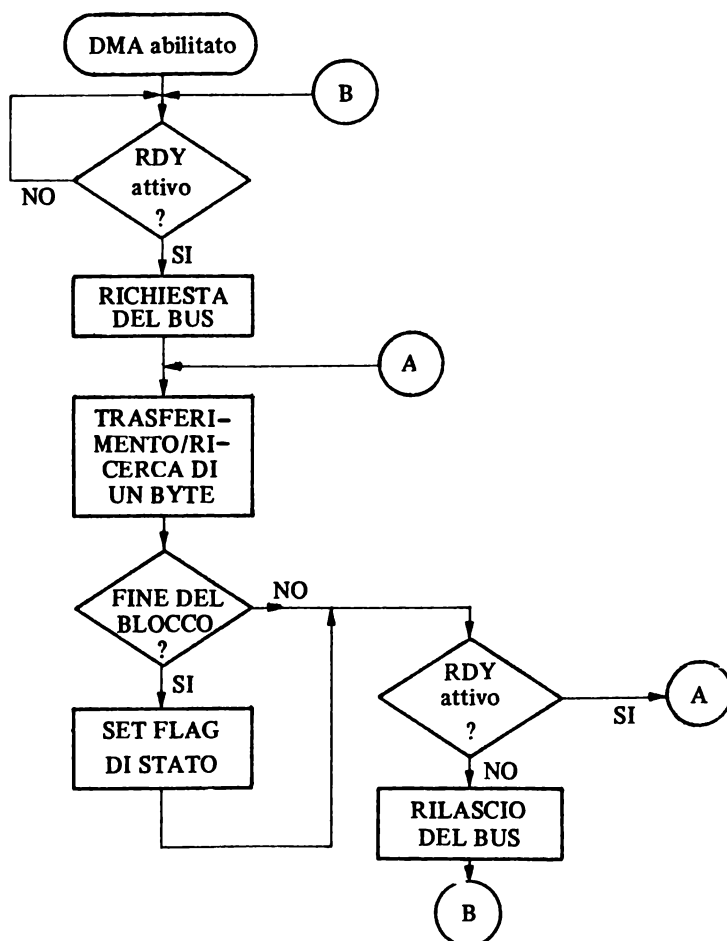



Fig. 9.15
Trasferimento a "pacchetti di byte".

Se il segnale RDY diventa inattivo prima della fine del trasferimento viene rilasciato il controllo del bus ed il DMAC si mette a testare lo stato di RDY per controllare una sua successiva attivazione e per procedere quindi ad un nuovo trasferimento una volta ottenuto nuovamente il controllo del bus.
Tale sequenza di operazioni continua fintantochè non è stato trasferito l'intero blocco di dati. Questo modo di operare presenta parecchi vantaggi: infatti impegna il bus solamente quando effettivamente si hanno dei dati disponibili da trasferire e quindi si può aumentare la velocità di trasferimento rispetto al caso precedente.
L'inconveniente è che, se si deve trasferire un blocco di dati piuttosto lungo e questi sono tutti disponibili, una volta iniziato il trasferimento la CPU rimane bloccata per tutto il tempo necessario e ciò a volte non è accettabile.

9.4.4. *"Modo continuo"*

Un ultimo modo possibile di operare dello Z80-DMA prende il nome di modo continuo. In questo caso il DMAC, una volta avuto il controllo del bus non lo rilascia fino a quando non è completato il trasferimento dell'intero blocco, oppure, se esso effettua anche la ricerca di un byte, fino a quando questa operazione non ha dato esito positivo.
Il diagramma di flusso di questo modo di funzionamento è riportato nella fig. 9.16.
Come si può notare se il segnale RDY è disattivato il DMAC si mette in attesa senza però rilasciare il controllo del bus. Questo è il modo più veloce di trasferimento di dati in quanto la richiesta per il rilascio del bus è fatta una sola volta e il DMAC è poi sempre pronto a trasferire i dati non appena essi siano disponibili. Purtroppo però il modo continuo blocca il funzionamento della CPU per intervalli di tempo che possono risultare eccessivi in certe applicazioni. Esso è utilizzato di solito quando si richiede un trasferimento di dati particolarmente veloce, e quando l'arresto dell'attività della CPU può essere tollerato.
Si è già detto che in tutti i tipi di funzionamento è possibile programmare il DMAC di procedere anche ad una ricerca di una certa configurazione di un byte. Una volta che questa ricerca ha dato esito positivo si possono richiedere azioni diverse da parte del DMAC. Si può ad esempio imporre che il trasferimento o la ricerca si arrestino e che quindi il DMAC rilasci il bus: la CPU può allora passare ad una lettura dei registri di stato del DMAC per dedurre la causa del rilascio, ed iniziare le azioni del caso. Questo modo di procedere ha l'inconveniente che la CPU deve indagare sui vari tipi di bit di stato del DMAC prima di dedurre la causa del rilascio: per questa indagine occorre ovviamente un certo tempo ed una certa organizzazione del software.

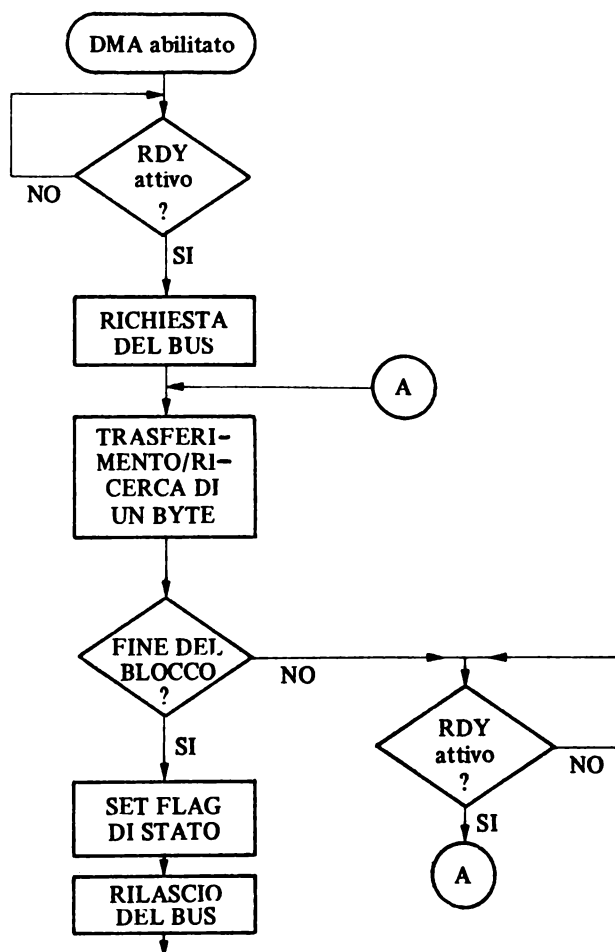



Fig. 9.16
Funzionamento del DMA in modo continuo.

Si può anche richiedere che, una volta trovato il byte, si arrestino immediatamente le operazioni di trasferimentro in corso, sia rilasciato il controllo del bus e sia inoltre effettuata una richiesta di interruzione che con il modo 2 di funzionamento dello Z80 permette anche di mandare in esecuzione immediatamente la relativa routine di servizio. Ciò è possibile in quanto lo Z80-DMA è in grado di fornire un vettore che può variare a seconda della causa che ha generato l'interruzione. In questo caso la CPU non deve procedere ad alcuna analisi dello stato del DMAC e quindi il tempo di attesa prima di iniziare la routine di servizio può essere notevolmente ridotto. E' anche possibile un altro modo di operare. Il DMAC, una volta trovato il byte cercato, prosegue il trasferimento fino al completamento del blocco e solo alla fine dà una segnalazione di interruzione alla CPU.

9.5. Flessibilità di interfacciamento dello Z80-DMA

Esistono diverse situazioni a causa delle quali il DMAC può richiedere interruzione alla unità centrale; occorre però tener presente che queste richieste possono essere onorate solamente quando il DMAC funziona da slave, in quanto, in caso contrario, la CPU non è in grado neppure di accorgersi dell'avvenuta richiesta.
Il DMAC può generare una richiesta di interruzione:

- su attivazione dell'ingresso READY
- qualora abbia trovato un dato avente la configurazione cercata, nelle operazioni di ricerca;
- alla fine del trasferimento di un blocco di dati.

Nel primo caso, cioè quando la linea esterna READY diventa attiva, il DMAC non ha ancora fatto richiesta del controllo del bus, mentre negli altri due casi esso sta svolgendo la funzione di master. Perché una richiesta di interruzione sia accettata dalla CPU è necessario che il DMAC abbia rilasciato il controllo del bus. Ognuna delle cause di interruzione setta un particolare bit in un registro di stato interno che la CPU può leggere e quindi dedurre la causa di interruzione. E' anche possibile far si che il DMAC fornisca il vettore di interruzione di valore diverso a seconda della causa che l'ha provocata e quindi utilizzare la potenzialità offerta dallo Z80 funzionante nel modo 2 per la gestione delle interruzioni.
Esistono altre caratteristiche di questo componente che consentono di semplificare non solo la struttura dell'hardware per un trasferimento di dati, ma anche quella del software.
Come già detto il DMAC deve essere opportunamente inizializzato per poter funzionare nel modo desiderato: occorre gli siano forniti gli indirizzi iniziali della sorgente e della destinazione, il numero di byte da trasferire ed altre indicazioni come ad esempio il modo di funzionamento richiesto, ecc.
Alla fine del trasferimento di un blocco di dati le diverse informazioni iniziali, come gli indirizzi e il numero di byte da trasferire, risultano cambiati; se si volesse far eseguire un successivo trasferimento con le stesse modalità, sarebbe necessario ripristinare i valori così alterati. Invece di procedere ad una nuova programmazione del DMAC si può utilizzare la caratteristica di funzionamento in "autorestart": una volta programmato il DMAC in modo da rendere attiva questa caratteristica, alla fine del trasferimento di un blocco di dati sono automaticamente ripristinati i diversi parametri così che si può effettuare un successivo trasferimento con le stesse caratteristiche del precedente.
Una tipica applicazione di questa possibilità si ha nel caso che la periferica sia costituita da un tubo a raggi catodici il quale deve essere periodicamente rinfrescato. Stabilita un'area di memoria da trasferire si ha con il DMAC

il rinfresco automatico senza nessuna necessità di intervento da parte della CPU. Si noti che durante il trasferimento possono essere variati alcuni parametri: ad esempio si supponga che si debbano trasferire differenti blocchi di dati, tutti della stessa lunghezza, in uscita; durante il trasferimento del primo blocco, non appena la CPU ha il controllo del sistema essa può alterare l'indirizzo iniziale della sorgente. Alla fine del primo trasferimento non è necessario anche in questo caso riprogrammare completamente il DMAC: con la funzione di autorestart esso attua un nuovo trasferimento prelevando i dati da un'area di memoria con indirizzo di partenza diverso mentre per tutte le altre caratteristiche il trasferimento procede come nel primo caso.
Durante il trasferimento di dati, essendo il DMAC master, esso non ha la possibilità di inviare alcuna informazione relativa al procedere dell'operazione, verso la CPU. Inoltre le periferiche esterne non sono in grado di leggere i registri di stato del DMAC anche perchè si trovano sicuramente nello stato di slave e quindi impossibilitate ad attivare una tale operazione di lettura.
D'altra parte può essere necessario conoscere lo stato di avanzamento del trasferimento in modo da poter intraprendere determinate azioni al verificarsi di certe situazioni, come ad esempio quelle che indicano un determinato numero di dati trasferiti.
Allo scopo è predisposto un segnale di uscita del DMAC che consente di fornire una tale informazione: la linea utilizzata è quella che serve anche per generare le richieste di interruzione INT in quanto sicuramente, durante un trasferimento, essa non può servire per questo scopo.
Si può predisporre un adatto contatore interno affinché questo segnale sia reso attivo dopo che è stato trasferito un numero di byte compreso tra 1 e 255. Dopo il primo impulso generato da questa linea, i successivi si ottengono ogni 256 dati trasferiti.

9.6. Caratteristiche dinamiche

Il funzionamento dinamico dello Z80-DMA può essere considerato uguale a quello del microprocessore Z80 nel senso che i diagrammi temporali dell'operazione di lettura o scrittura in memoria oppure in un dispositivo di I/O sono perfettamente identici a quello dello Z80.
Si consideri ad esempio il diagramma temporale relativo al trasferimento di un dato dalla memoria ad un dispositivo di uscita, indicato nella fig. 9.17, ove si suppone non siano stati inseriti periodi di attesa, Tw.
La prima fase del trasferimento cioè la lettura della memoria, è uguale all'omonimo ciclo macchina della CPU Z80 tanto che la memoria non ha nessuna possibilità di distinguere se l'operazione di lettura è comandata dalla CPU o da un altro dispositivo. I dati letti sul fronte di discesa del

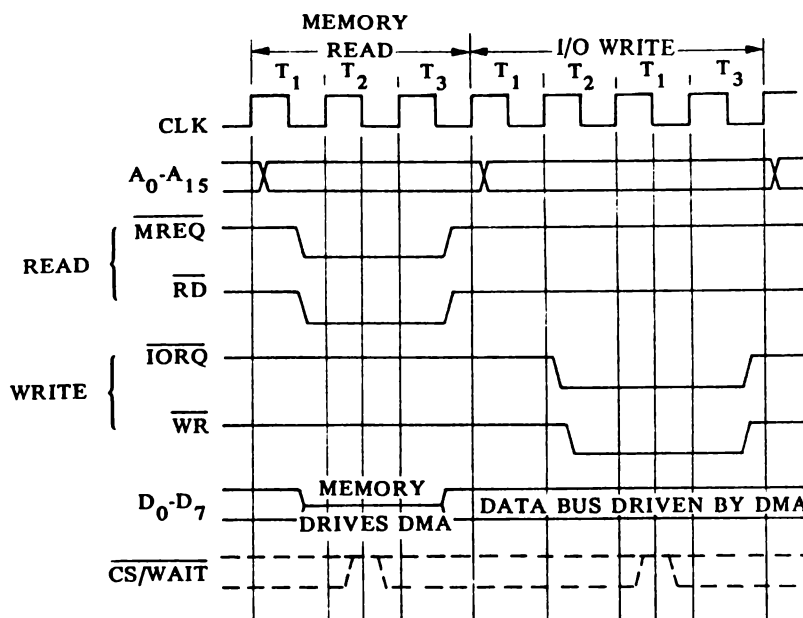



Fig. 9.17
Diagramma temporale di un trasferimento sequenziale da memoria a porta di uscita.
(Zilog, Z80-DMA Technical Manual)

clock nel periodo T3 sono mantenuti disponibili dal DMAC sul bus dei dati per la successiva operazione di scrittura su un dispositivo di uscita. Anche durante questa seconda fase il comportamento è identico a quello che si ha durante un ciclo macchina controllato dalla CPU; l'unica differenza, non sostanziale, è il fatto che i dati sono ora disponibili sin dall'inizio del ciclo e non solo dopo il fronte di discesa del segnale di clock, durante T1, come accade con la CPU.
Occorre anche far notare che possono essere inseriti degli stati di attesa durante le due operazioni di lettura e di scrittura; questa inserzione ovviamente rallenta la velocità di trasferimento dei dati. Per l'attivazione del segnale WAIT, che provoca l'inserzione degli stati di attesa, è necessario un circuito esterno aggiuntivo, già a suo tempo illustrato.
Il DMAC della famiglia Z80 presenta inoltre la interessante caratteristica di potere variare i cicli di lettura o di scrittura, sia relativi alla memoria che alle porte di ingresso e di uscita. Ciò è ottenibile con una opportuna programmazione che permette non solo di allungare tali cicli ma anche di renderli più brevi rispetto a quello tipico della CPU permettendo così di ottenere la massima velocità di trasferimento consentita.

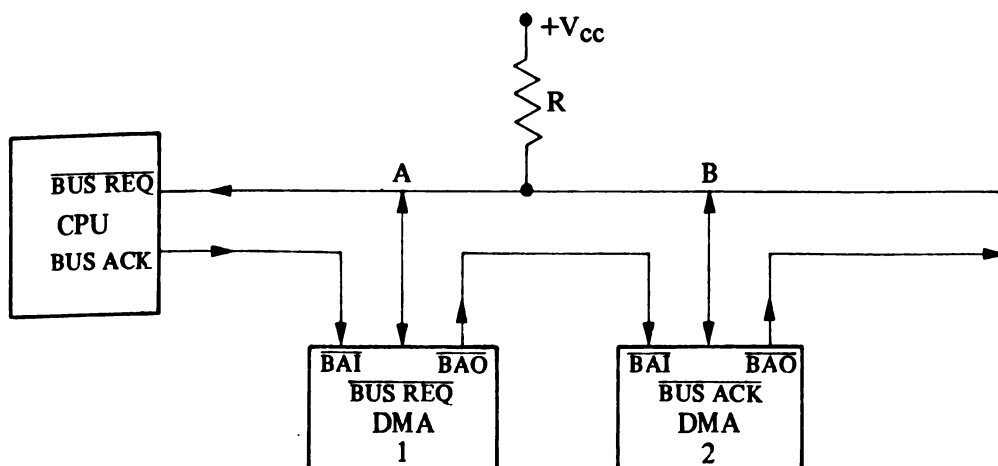



Fig. 9.19
Collegamento in daisy-chain di più DMAC.

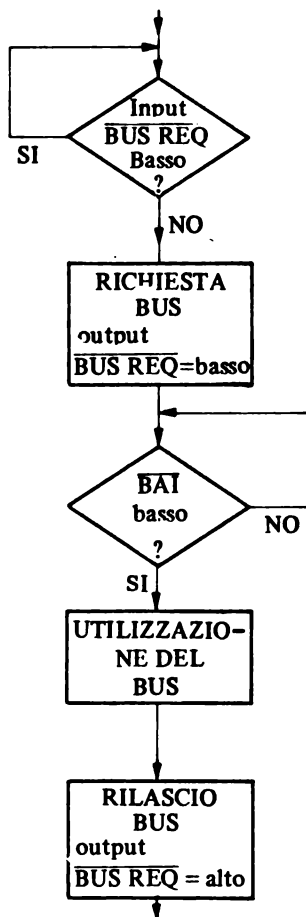



Fig. 9.20
Procedura per la richiesta del bus in un sistema con più DMAC.

E' perciò possibile adattare alle diverse esigenze il numero di periodi di clock che compongono un ciclo di lettura o di scrittura: questi cicli possono durare 2, 3 oppure 4 periodi di clock.
Inoltre si può imporre che i vari segnali di comando utilizzati nel trasferimento si disattivino con mezzo periodo di clock di anticipo, come indicato in fig. 9.18.

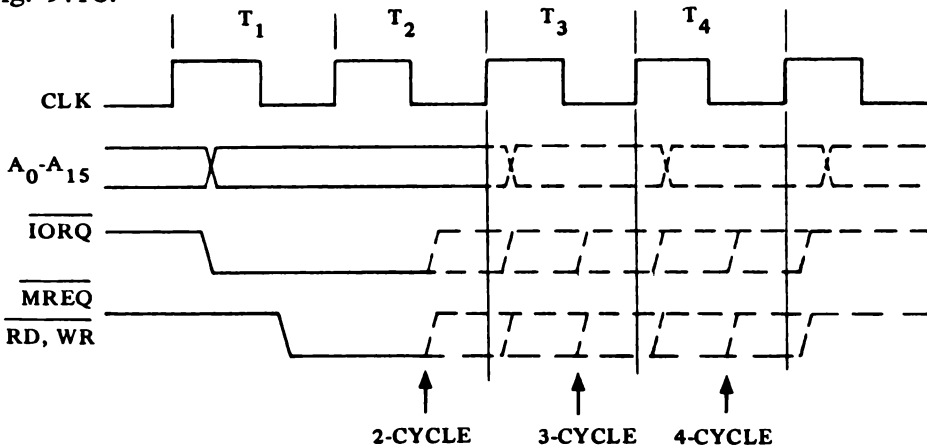



Fig. 9.18
Le possibilità di variare la durata dei segnali di controllo generati dallo Z80-DMA (Zilog, Z80-DMA Technical Manual)

9.7. Sistemi con più DMAC

Quando in un sistema sono utilizzate diverse unità periferiche che richiedono un trasferimento veloce di dati è conveniente ricorrere all'impiego di più DMAC. L'utilizzazione di uno solo infatti richiederebbe una frequente riprogrammazione del componente ed una adeguata struttura hardware per la selezione delle diverse porte di I/O, ciò che ridurrebbe la convenienza dell'utilizzazione del DMAC stesso.
Con l'impiego di più DMAC si pone il problema della risoluzione delle priorità quando alcuni di essi richiedono il controllo del bus contemporaneamente. Con lo Z80-DMAC è possibile risolvere questo problema senza la necessità di utilizzare dell'hardware aggiuntivo: lo schema di collegamento adottato è illustrato nella fig. 9.19.
Ogni DMAC possiede tre linee per la gestione automatica delle richieste dei bus: BAI/ BAO/ e BUSREQ/. La linea BAI/ è di ingresso, la BAO/ è di uscita e BUSREQ/ è invece bidirezionale. La procedura seguita è quella illustrata in fig. 9.20.

La linea BUSREQ/, se il DMAC è nello stato di slave, è una linea di ingresso. Una volta che sia attivata la condizione, interna o esterna, per la richiesta del controllo del bus, il DMAC interessato analizza la linea BUSREQ/: solo se questa si trova a livello logico alto, cioè se non è attiva, esso può effettuare la richiesta alla CPU ponendo a livello logico basso tale linea. La CPU informa successivamente di avere abbandonato il bus del sistema portando la linea BUSACK/ a livello logico basso. Tale linea è collegata a BAI/ (bus acknowledge input) del primo DMAC della catena, che ha perciò la più elevata priorità: solo se non ha fatto richiesta di controllo del bus, questo DMAC pone a livello logico basso la linea di uscita BAO/ (Bus Acknowledge output) che a sua volta è collegato al segnale BAI/ del DMAC successivo. Se ad esempio era stato il DMAC n. 2 ad avere fatto la richiesta, esso può prendere il controllo del bus, mantenendo disattivato il rispettivo segnale BAO/ fintantoché non rilascia i bus. Si noti che il segnale BUSREQ/ è costantemente tenuto a livello attivo: in queste condizioni nessun dispositivo della catena può richiedere il controllo dei bus se questi non sono stati precedentemente rilasciati dal dispositivo n. 2.
Al rilascio del controllo del Bus il DMAC n. 2 rilascia la linea BUSREQ/ la quale va a livello logico alto mentre l'uscita BAO/ è posta al valore basso.
E' ora possibile sia effettuata una richiesta di rilascio del bus anche da parte di altri dispositivi sia a priorità maggiore che minore. Può accadere che più DMAC facciano contemporaneamente la richiesta del controllo del bus: ciò è possibile in quanto nell'intervallo di tempo che intercorre dall'istante in cui uno qualsiasi di essi controlla lo stato della linea BUSREQ/ a quello in cui esso effettivamente la attiva, è possibile che anche altri dispositivi procedano ad una analoga operazione. In tale situazione la linea BUSREQ/ è quindi tenuta a livello logico basso per effetto di più dispositivi: quando l'unità centrale rilascia il controllo del bus esso porta a livello basso BUSACK/ e questo segnale si propaga lungo la catena. Il primo dispositivo nella catena, che ha fatto richiesta, quando trova il suo ingresso BAI/ a livello logico basso prende effettivamente il controllo del bus; non appena lo rilascia però, il suo segnale di uscita BAO/ è posto a livello basso per cui nella catena si propaga il rilascio ed il controllo viene assunto dal dispositivo a priorità immediatamente inferiore.
Da quanto detto si può notare che tale modo di procedere è analogo a quello che si ha nella gestione delle interruzioni nel caso in cui si riabiliti la CPU, ad accettare richieste, solo alla fine di una routine di servizio.

9.8. Fly-by

Lo Z80-DMA può anche essere utilizzato nel modo fly-by o simultaneo; in questo modo il DMAC effettua solo la lettura dei dati dalla sorgente di in-

formazioni demandando ad un hardware opportuno la contemporanea scrittura del dato nel dispositivo di destinazione.
Un possibile schema atto al fly-by è quello indicato in fig. 9.21, dove i dispositivi interessati sono la memoria ed una porta di I/O.
Il DMAC può trasferire dati dalla memoria alla porta di I/O o viceversa, utilizzando un solo ciclo di lettura per ogni dato da trasferire.
Sono utilizzati solamente due segnali di controllo del DMAC quando si trova nello stato di master e precisamente RD/ e IORQ/; il DMAC attiva in ogni caso solo l'operazione di lettura: se IORQ/ si trova a livello logico alto la lettura è relativa alla memoria ed automaticamente è attivata una operazione di scrittura sul dispositivo di I/O. Se viceversa IORQ/ è a livello logico basso la lettura è relativa ad una periferica e la scrittura avviene sulla memoria.
Da RD/ e IORQ/ devono allora essere ricavati tutti i segnali di controllo che servono per le operazioni relative alla sorgente e alla destinazione. Allo scopo viene utilizzato un multiplexer, come indicato nella fig. 9.21, il quale è attivo solamente quando il DMAC ha assunto il controllo del bus mentre in caso contrario le sue uscite si trovano nello stato di alta impedenza. I quattro segnali di uscita dal multiplexer assumono valori diversi in funzione del valore del segnale di selezione, che coincide con IORQ/. In tabella 9.1 sono riportati tali valori ed i significati assunti dalle varie uscite:

Tabella 9.1

SEL	RD	Y_1 (MRD)	Y_2 (MWR)	Y_3 (IORD)	Y_4 (IOWR)	
1	0	0	1	1	0	lettura da memoria
0	0	1	0	0	1	lettura da I/O

Significato dei segnali generati dal Multiplexer di fig. 9.21.

Nella fig. 9.22 è riportato il diagramma temporale di un trasferimento da memoria alla porta di I/O in modo continuo oppure a burst.
Si noti che sono attivati contemporaneamente i due segnali MRD/ e IOWR/. La memoria deve essere in grado di fornire i dati con una certa velocità e questi vengono acquisiti dalla periferica sul fronte di salita di IOWR/, che ha lo stesso andamento temporale del segnale MRD/.
Quando la CPU vuole accedere alla memoria o alla porta di I/O i due segnali RD/ e WR/ da essa generati sono inviati sia alla memoria sia alla periferica in quanto in questo caso BUSACK/ non è attivo, per cui mentre il multiplexer ha le sue uscite nello stato di alta impedenza, sono invece attivi i due buffer A e B di fig. 9.21.

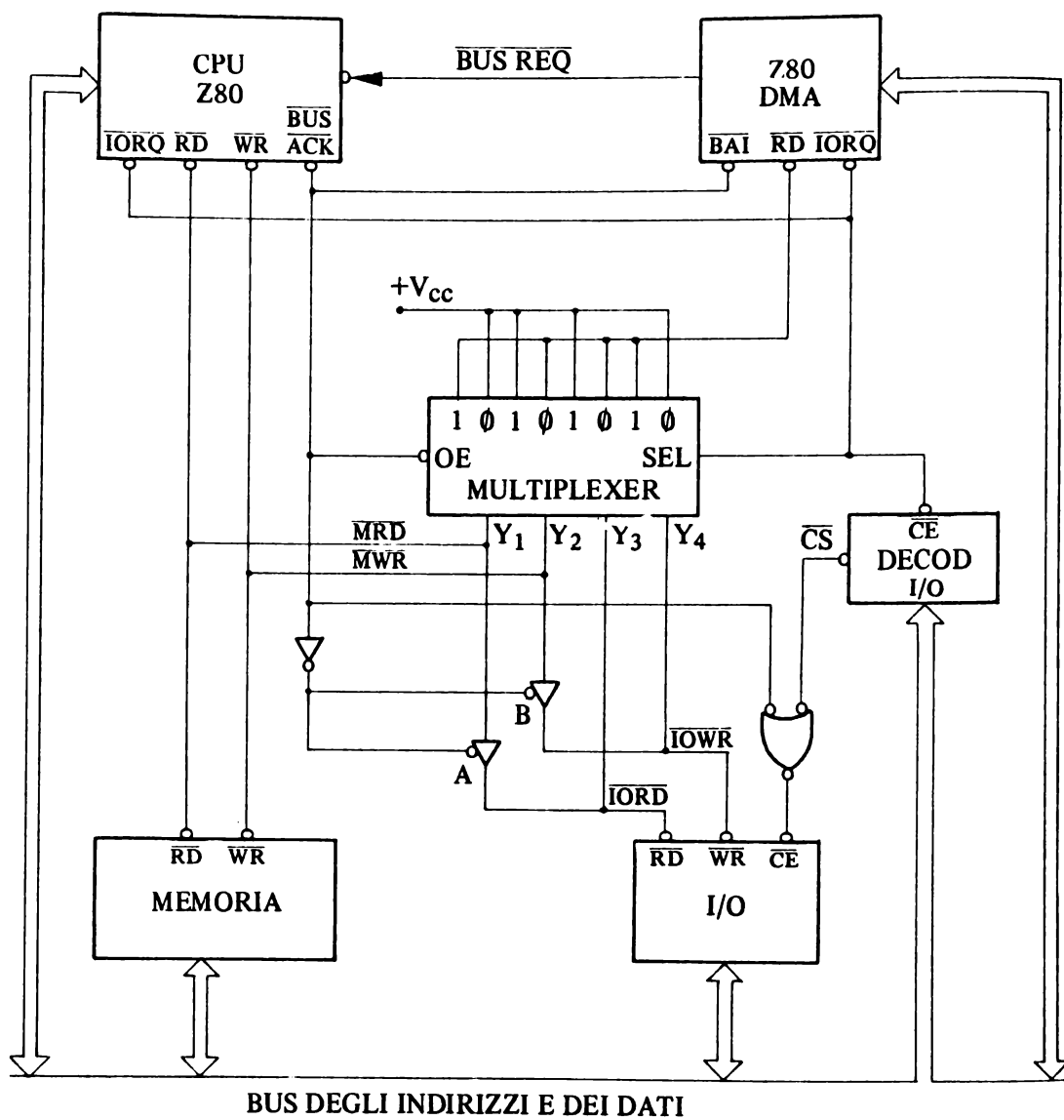



Fig. 9.21
Schema per l'utilizzazione dello Z80-DMA in modo "fly-by".

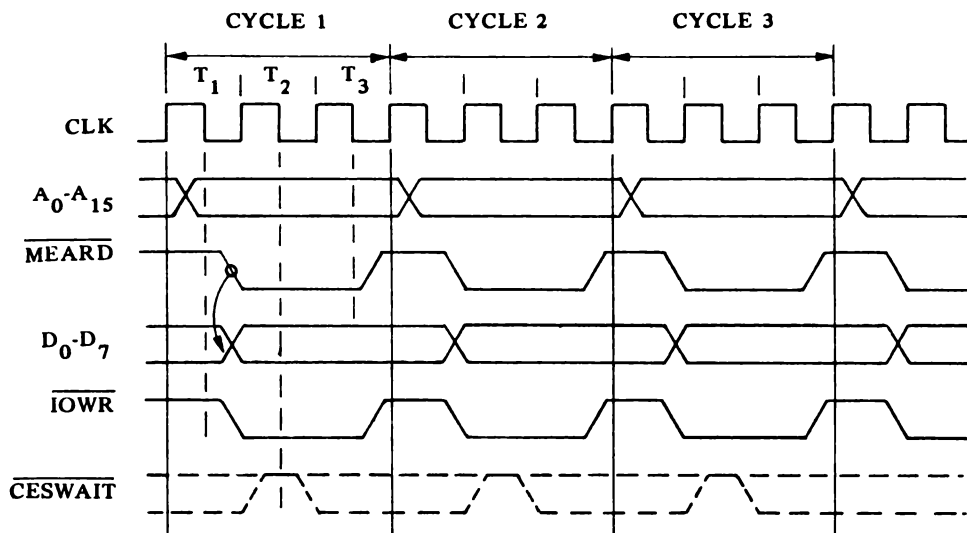



Fig. 9.22
Diagramma temporale di trasferimento continuo e a burst fra memoria e I/O (Zilog, Z80-DMA Technical Manual)

Resta ancora da analizzare nello schema considerato l'attivazione del segnale CS/ relativo alla sorgente oppure alla destinazione. Per quanto riguarda la memoria non ci sono problemi e si possono adottare le varie tecniche illustrate in precedenza. Per la periferica invece si devono tener presenti alcune caratteristiche di questo modo di funzionamento. Si deve infatti notare che il DMAC fornisce solamente un indirizzo relativo alla sorgente, formato da 16 bit. Il decodificatore degli indirizzi delle porte di I/O deve quindi presentare delle caratteristiche particolari sia perché l'indirizzo è composto da 16 bit, sia perché continuamente varia al variare della posizione di memoria interessata.

9.9. Uso del DMA con il SIO

Molto di frequente le periferiche che richiedono una certa velocità di trasferimento di dati inviano e ricevono questi in forma seriale. E' quindi necessaria la conversione da serie a parallelo e viceversa. Allo scopo può essere impiegato uno Z80-SIO assieme ad uno Z80-DMA.
Uno schema a blocchi estremamente semplificato può essere quello indicato nella fig. 9.23.
In questo caso si deve procedere ad una adatta programmazione sia del DMAC che del SIO. In particolare è utilizzato il segnale di RDY generato dal SIO per indicare quando deve intervenire il DMAC per eseguire il tra-

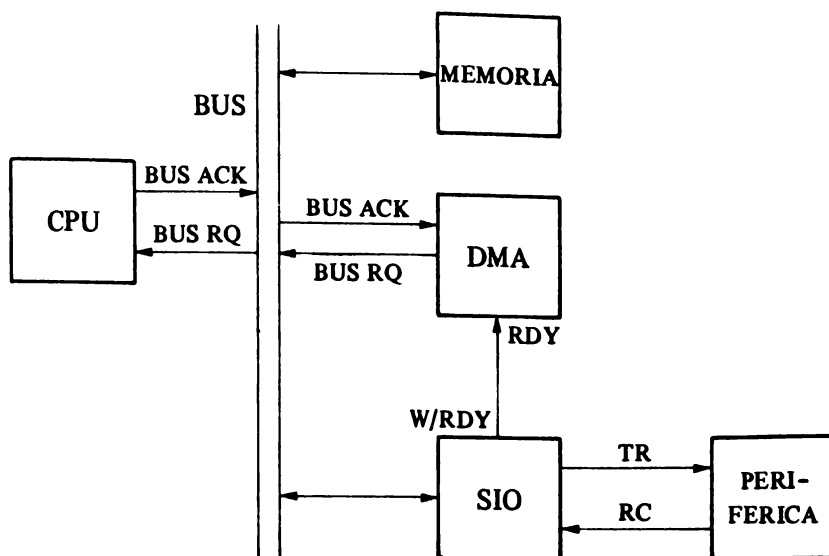



Fig. 9.23
Schema a blocchi per l'impiego di un DMAC e di un SIO.

sferimento di dati. Poiché questo segnale può essere attivato solamente o in ricezione oppure in trasmissione ma non contemporaneamente, è necessaria una adeguata organizzazione software per la gestione di periferiche che richiedono trasferimenti nelle due direzioni.
Un modo di procedere potrebbe essere quello di predisporre sempre il SIO in ricezione e per eseguire invece una trasmissione procedere ad una programmazione preliminare e poi ritornare, alla fine, alla situazione di partenza. Se si ha il SIO programmato in modo che RDY si attiva in ricezione, quando ciò accade viene richiesto l'intervento del DMAC che a sua volta richiede il controllo del bus e procede al trasferimento. Nel caso invece si voglia eseguire una trasmissione, poichè questa viene sempre intrapresa dalla CPU si procede ad una modifica della programmazione del SIO in modo che RDY si attivi in trasmissione. La CPU dà inizio quindi alla trasmissione. Il SIO richiede i dati attivando RDY e quindi li acquisisce tramite un'operazione di trasferimento organizzato dal DMAC. Alla fine del trasferimento è necessario procedere ad una nuova variazione del funzionamento del SIO con ulteriori comandi in modo da predisporlo per la ricezione.

Capitolo 10

LA GESTIONE DI VISUALIZZATORI E TASTIERE

10.1. Generalità

In un elaboratore l'interazione con l'operatore avviene di solito per il tramite di telescriventi o di terminali video i quali, con la loro flessibilità, rendono agevoli sia l'immissione di dati o di comandi sia la lettura dei risultati ottenuti. Spesso però, specialmente in sistemi a microprocessori, sono sufficienti delle periferiche di ingresso o di uscita meno sofisticate. Può bastare infatti in questi casi una semplice tastiera, con un ridotto numero di tasti, ed un dispositivo di visualizzazione capace di fornire un certo numero, anch'esso piccolo, di caratteri.
In questo capitolo si analizzeranno i metodi di interfaccia di tastiere e di visualizzatori.

10.2. Visualizzatori

Il dispositivo attualmente più utilizzato, a causa del suo basso costo, per ottenere delle segnalazioni luminose è il diodo ad emissione di luce, LED (Light Emitter Diode); esso emette delle radiazioni luminose quando è percorso da una corrente di sufficiente intensità. A seconda del materiale semiconduttore utilizzato sono emesse radiazioni luminose di lunghezze d'onda diverse e quindi di diversi colori: sono disponibili led rossi, verdi o gialli; i più usati, per l'efficienza di conversione, sono quelli che emettono luce rossa.
Lo schema di principio per l'utilizzazione di un LED è quello riportato in fig. 10.1.
Quando il diodo è conduttore, con una tensione V_d ai suoi capi di circa 1,5 V, la corrente che lo percorre è tipicamente dell'ordine di qualche decina di mA.
Supponendo che la tensione V_{cc} sia pari a 5 V il valore della resistenza R da inserire nel circuito per avere la circolazione di una corrente di 20 mA si calcola facilmente:

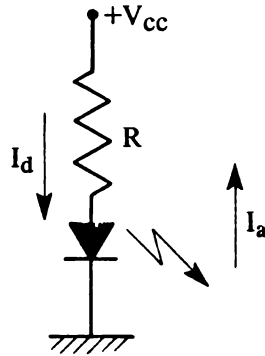



Fig. 10.1
Schema di principio per l'utilizzazione di un LED.

$$R = \frac{V_{cc} - V_d}{I_d} = \frac{5 - 1,5}{20 \times 10^{-3}} \cong 180\ \Omega$$

Per pilotare un tale componente, in un sistema a microprocessore, è necessario utilizzare una porta di uscita che sia in grado di fornire la corrente richiesta; uno schema potrebbe essere quello indicato in fig.10.2, dove si ha l'accensione del LED quando si invia un valore logico alto in uscita.

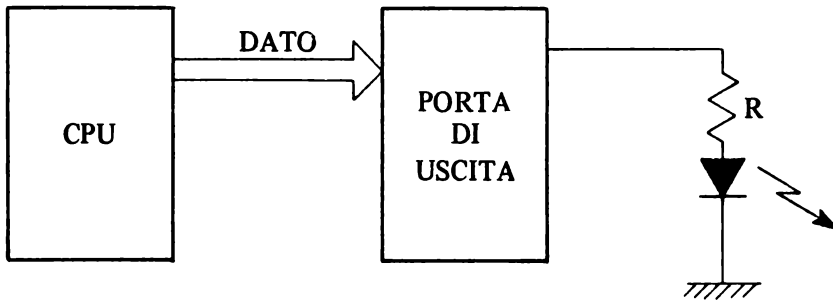



Fig. 10.2
Alimentazione di un LED tramite una porta di uscita (logica positiva).

In qualche caso potrebbe essere conveniente invece avere la situazione opposta e cioè ottenere l'indicazione luminosa quando l'uscita della porta è a valore logico basso. Si potrebbe allo scopo inserire un invertitore all'uscita della porta ma si può anche utilizzare lo schema riportato in fig. 10.3. Nel caso che la porta utilizzata per il comando del LED non sia in grado di fornire o assorbire la corrente necessaria, si deve interporre un circuito

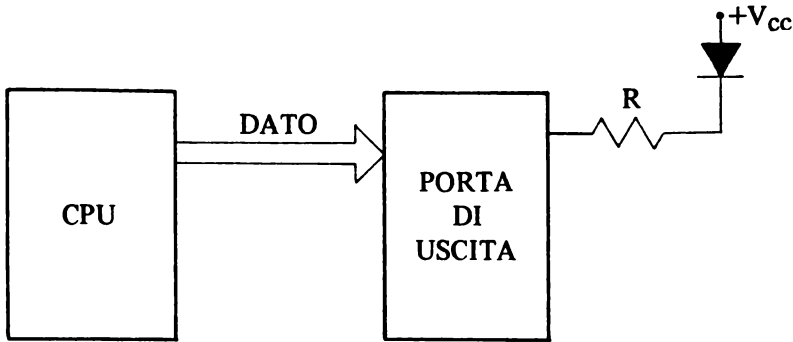



Fig. 10.3
Alimentazione di un LED tramite una porta di uscita (logica negativa).

amplificatore di corrente: un esempio è riportato nella fig. 10.4, dove a sinistra ci si riferisce al funzionamento in logica positiva, mentre a destra a quello in logica negativa.

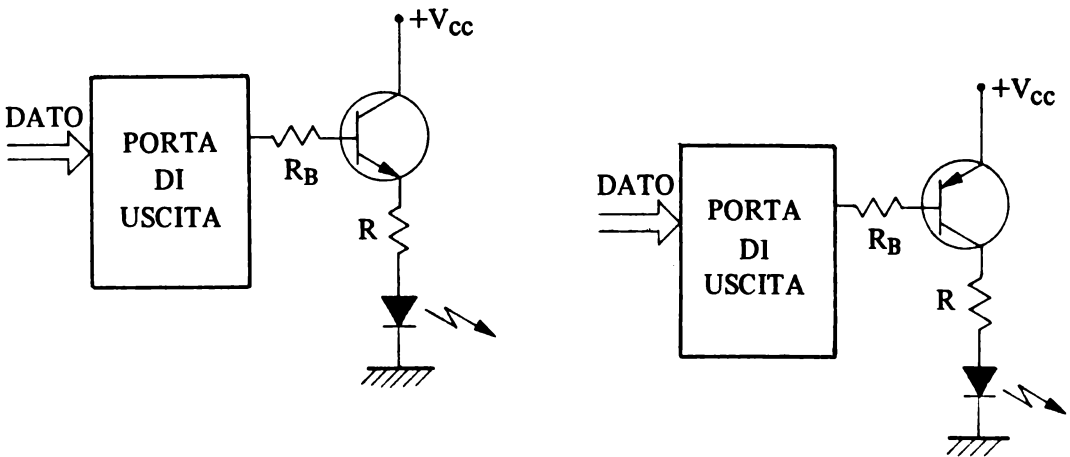



Fig. 10.4
Inserzione di un amplificatore per il comando di un LED.

L'informazione ottenibile con un semplice LED è ovviamente limitata ad un solo bit e quindi è utilizzabile solo in certi tipi di applicazione.
Per ottenere delle informazioni di tipo numerico si possono utilizzare visualizzatori (o display): essi sono costituiti da 7 diodi disposti come in fig. 10.5.

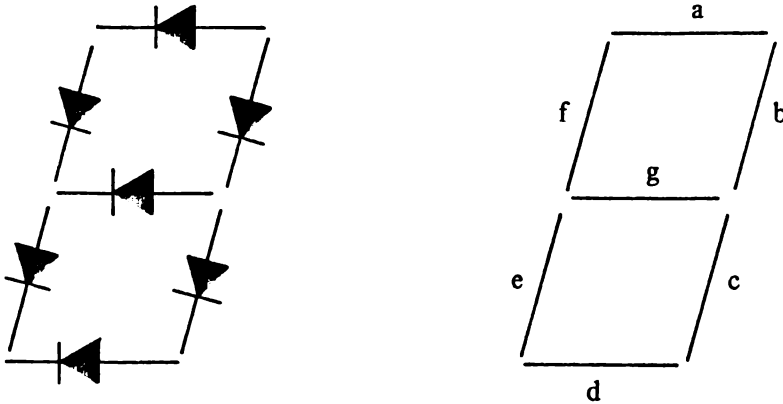



Fig. 10.5
Visualizzatore numerico a LED a 7 segmenti.

L'attivazione di un diodo rende luminoso il corrispondente segmento del display per cui con un pilotaggio opportuno dei diodi è possibile visualizzare in forma stilizzata tutte le 10 cifre, come riportato in fig. 10.6.

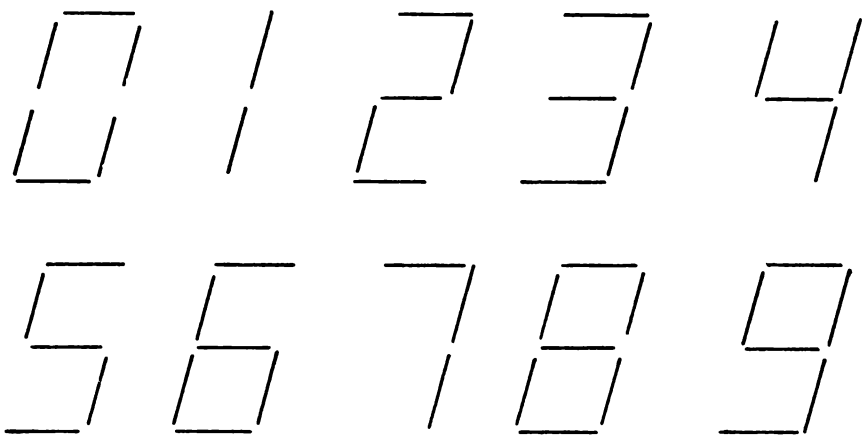



Fig. 10.6
Cifre visualizzate da un LED a 7 segmenti.

Con display a 7 segmenti è anche possibile visualizzare alcune lettere dell'alfabeto come, ad esempio A C H P E F b d; si osservi che se si accetta che le due lettere b e d siano rappresentate in carattere minuscolo e le lettere A C E F in maiuscolo si può ottenere da un display a sette segmenti una rappresentazione numerica in codice esadecimale, molto utilizzata in

certe applicazioni. Per visualizzare tutti i caratteri alfanumerici si deve ricorrere ad altri tipi di visualizzatori, che presentano un maggior numero di segmenti, o che hanno i diodi LED disposti a matrice, come sarà in seguito illustrato.
Nel display a sette segmenti si possono avere due disposizioni circuitali dei diodi e precisamente questi possono avere tutti gli anodi, oppure tutti i catodi, collegati fra loro, come è rappresentato nella fig. 10.7.

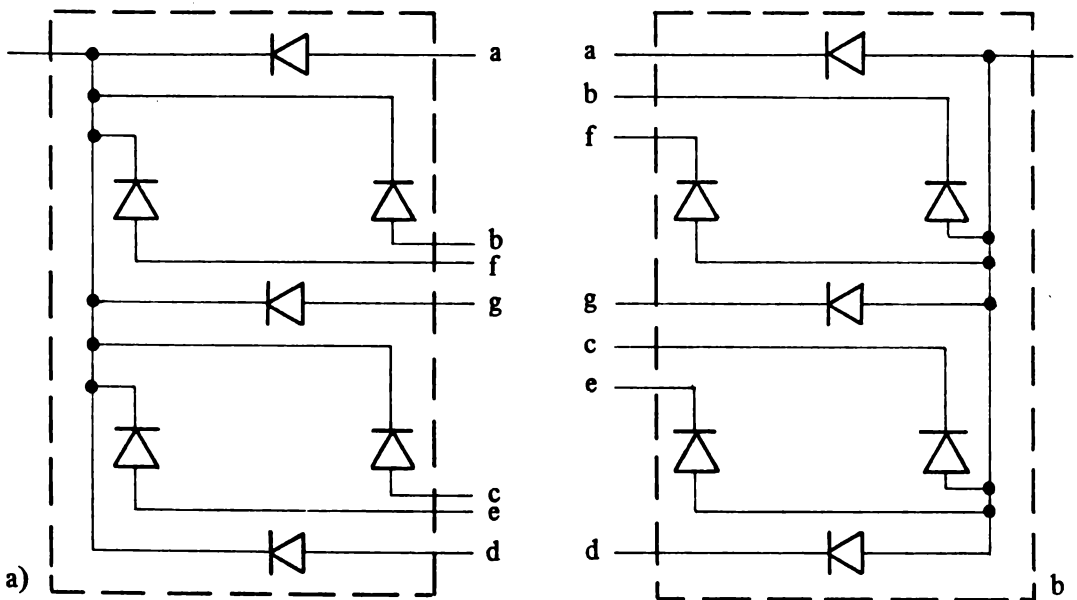



Fig. 10.7
Display a 7 segmenti a catodo (a) o ad anodo (b) comune.

10.2.1. *Collegamento di un visualizzatore ad un microprocessore*

Per ottenere la visualizzazione di una cifra è necessario siano attivati solo alcuni diodi del display a 7 segmenti: si deve, in altre parole, eseguire una codifica per passare dall'informazione numerica, che si ha a disposizione all'interno dell'elaboratore, a quella necessaria per ottenerne la visualizzazione.
Tale trasformazione di codici si può ottenere per via software con un opportuno programma di conversione ed allora è sufficiente realizzare la struttura hardware indicata nello schema di fig. 10.8.

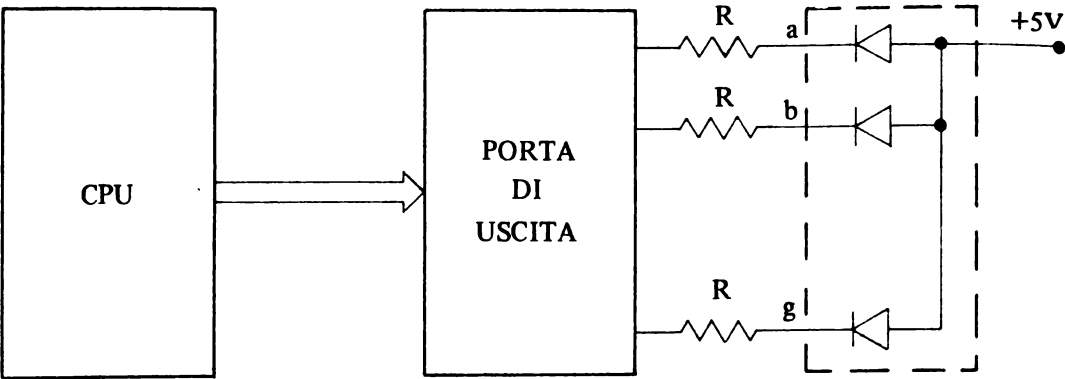



Fig. 10.8
Impiego di un display adottando una decodifica software.

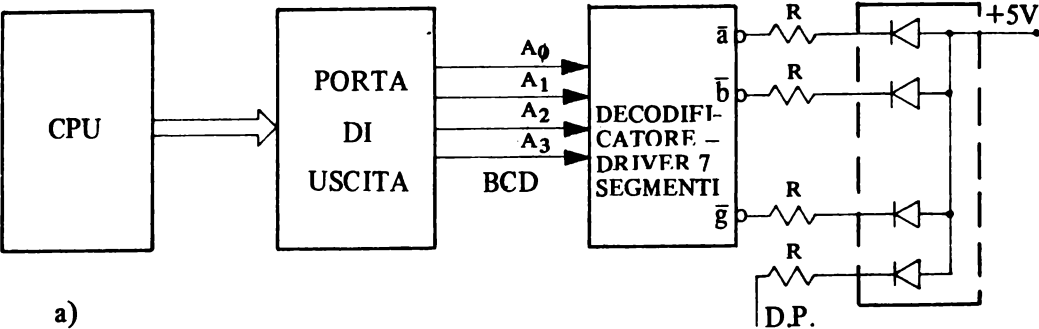
b)

$A_0\ A_1\ A_2\ A_3$	$\bar{a}\ \bar{b}\ \bar{c}\ \bar{d}\ \bar{e}\ \bar{f}\ \bar{g}$	DECIMAL
L L L L	H H H H H H H	OFF
L L L L	L L L L L L H	0
H L L L	H H H H L L H	1
L H L L	L L H L L H L	2
H H L L	L L L L H H L	3
L L H L	H L L H H L L	4
H L H L	L H L L H L L	5
L H H L	H H L L L L L	6
H H H L	L L L H H H H	7
L L L H	L L L L L L L	8
H L L H	L L L H H L L	9
L H L H	H H H H H H H	OFF
H H L H	H H H H H H H	OFF
L L H H	H H H H H H H	OFF
H L H H	H H H H H H H	OFF
L H H H	H H H H H H H	OFF
H H H H	H H H H H H H	OFF

Fig. 10.9
Impiego di un decodificatore BCD/7 segmenti (a) e sua tabella di verità (b).

La codifica su accennata può anche essere realizzata via hardware: esistono dei circuiti integrati adatti alla realizzazione dell'interfaccia per l'utilizzazione di un display; essi presentano le caratteristiche di fornire (od assorbire) la corrente richiesta dai diodi assieme a quella di memorizzare l'informazione loro fornita; ma soprattutto essi sono in grado di effettuare la codifica necessaria per l'utilizzazione del visualizzatore. In genere il microprocessore invia a tali componenti una informazione numerica in codice BCD o esadecimale ed essi la codificano nella forma necessaria per attivare correttamente i segmenti del display. In fig. 10.9 è indicato un possibile schema di impiego e la relativa tabella di conversione di un decodificatore da codice BCD a codici a 7 segmenti.

10.2.2. *Impiego del multiplexer*

Molto spesso è necessario avere a disposizione un certo numero di cifre al fine di ottenere l'indicazione di un valore numerico con un certo grado di precisione o per poter aumentare il campo dei valori che possono essere visualizzati.
Per il comando di questo insieme di display si adotta solitamente la tecnica del multiplexer, in quanto si ottiene una semplificazione nella struttura hardware necessaria.
Sono possibili infatti due modi di funzionamento dei display a stato solido; il primo consiste nell'applicare loro una tensione costante così che il diodo interessato emetta luce con continuità. Per operare in tale modo è necessario un codificatore per ogni display: una soluzione di questo genere viene adottata quando si debbano usare un numero limitato di display per cui il costo dei componenti necessari con la tecnica del multiplexer è superiore al costo dei decodificatori.
In realtà l'efficienza di un display è piccola, se viene alimentato con una corrente continua, rispetto a quella ottenuta se si utilizza un picco di corrente di opportuno valore e durata. Ovviamente con l'invio di una corrente impulsiva si avrà una discontinuità dell'intensità luminosa dei vari segmenti, ma, poiché l'occhio umano è un sensore relativamente lento, esso percepisce come continuo il segnale luminoso emesso, se la velocità di ripetizione degli impulsi è sufficientemente elevata. Se si organizza un circuito di rinfresco per ogni digit con una frequenza di 100 Hz o più si ha una visione continua del display. Tale tecnica presenta il vantaggio, a volte non trascurabile, di richiedere meno energia a parità di sensazione luminosa prodotta dai display.
Lo schema di principio del circuito che utilizza questa tecnica è indicato in fig. 10.10.

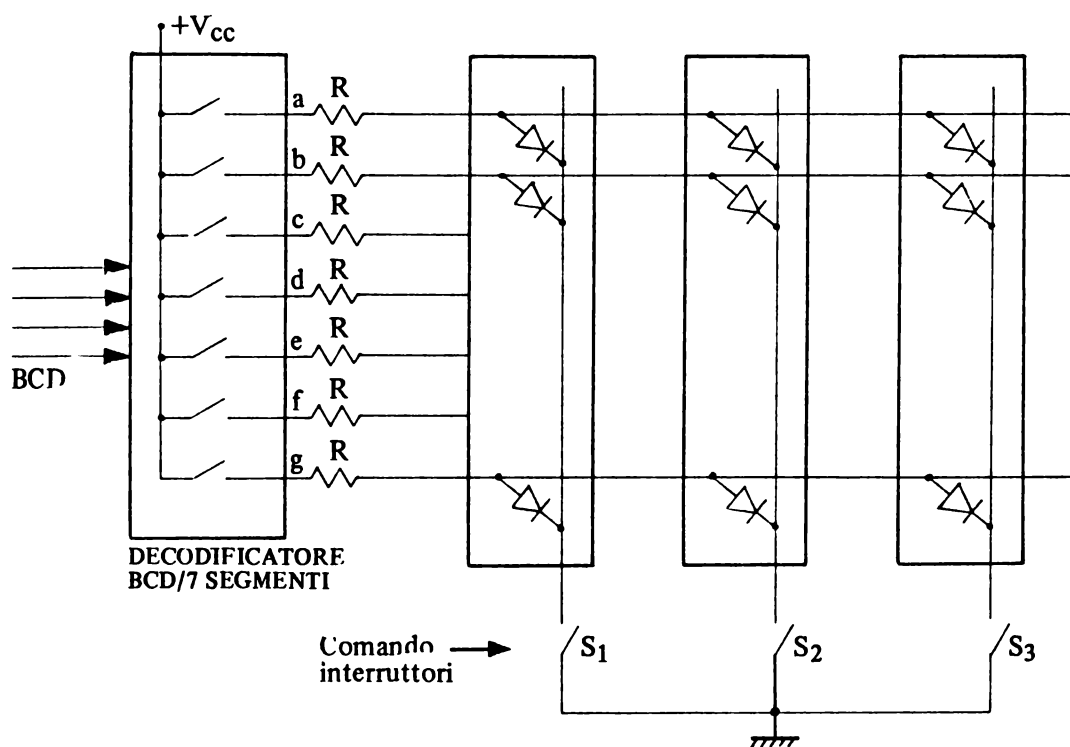



Fig. 10.10
Schema di principio per l'uso di display con la tecnica del multiplexing.

Il decodificatore BCD/7 segmenti alimenta solamente quelle linee (a-g) che servono per la visualizzazione di una determinata cifra. Tutti i diodi omonimi dei display sono collegati in parallelo alle linee che escono da questo decodificatore. Circola corrente nei diodi e quindi appare illuminato solamente quel display il cui interruttore S risulta chiuso. La corrente che circola su ogni segmento dipende oltre che dal valore della tensione di alimentazione anche da quello della resistenza R, che andrà scelta anche in funzione del picco di corrente che si desidera ottenere e dell'intervallo di tempo in cui si ha la conduzione. Per avere la presentazione delle varie cifre sui display è necessario chiudere in sequenza i vari interruttori in sincronismo con i dati presentati all'ingresso al decodificatore. Queste operazioni di sincronizzazione possono essere ottenute con il circuito di fig. 10.11, che per semplicità descrive il caso di quattro display.

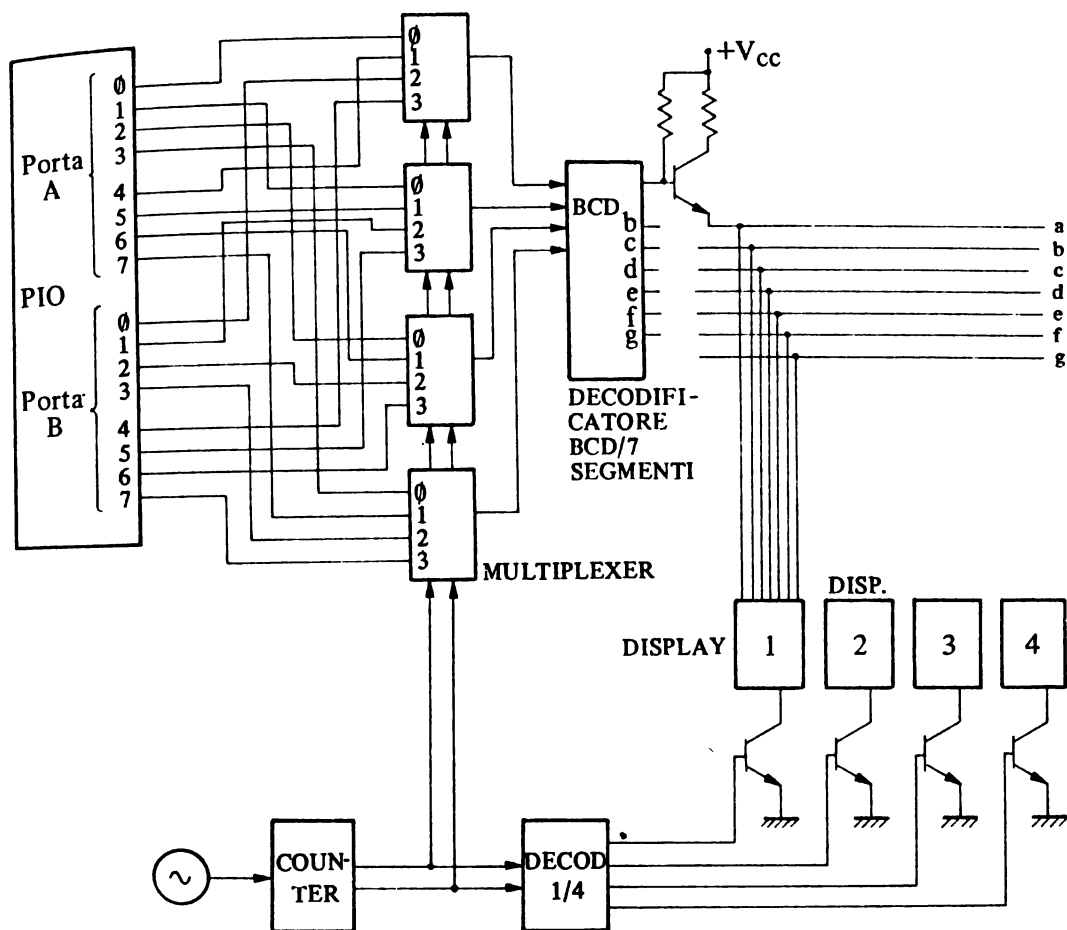



Fig. 10.11
Schema per l'impiego di 4 display a 7 segmenti con l'uso di un multiplexer.

In esso i codici BCD delle quattro cifre da visualizzare sono inviati dal microprocessore in una PIO, dove sono memorizzati; essi sono quindi presenti alle uscite delle porte A e B come indicato in fig. 10.12.

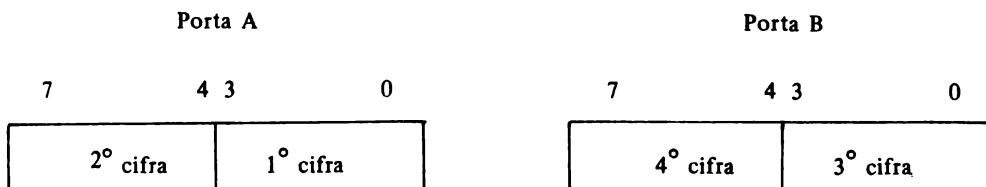



Fig. 10.12
Formato dei dati in uscita dalla porta PIO nello schema di fig. 10.11.

Un clock esterno invia degli impulsi ad un contatore, il quale fornisce in uscita il conteggio degli impulsi arrivati, con modulo 4, pari cioè al numero di display utilizzati.
Le due uscite del contatore sono inviate all'ingresso di un multiplexer il quale mette in comunicazione le uscite delle porte della PIO con il decodificatore BCD/sette segmenti. In tal modo all'ingresso del decodificatore si presentano in successione le quattro cifre da visualizzare. Un solo display alla volta è però selezionato, e quindi reso attivo, dal decodificatore 1/4 il quale ha gli ingressi collegati alle uscite del contatore. In tal modo si ha una univoca corrispondenza fra il dato selezionato dal multiplexer ed il display attivato.
Rispetto al caso in cui si utilizza un decodificatore per ogni display si hanno alcuni vantaggi ma anche qualche componente in più: si sono infatti utilizzati un multiplexer, un decodificatore ed un contatore.
Non è facile a priori stabilire quale è la tecnica più conveniente da utilizzare: infatti il numero di componenti da impiegare nel caso del multiplexer non cresce con l'aumentare del numero di digit da visualizzare ma contemporaneamente si deve inviare un picco di corrente sempre più grande per ottenere lo stesso effetto luminoso.
Si noti che nello schema di fig. 10.11 il microprocessore invia sulle porte di uscita solo quanto deve essere visualizzato ed è la parte hardware esterna che provvede al comando dei vari LED delle diverse cifre ed alla relativa operazione di rinfresco.

10.2.3. *Impiego di memorie*

Con la tecnica del multiplexer si visualizzano normalmente fino a 16 cifre in modo soddisfacente; in tal caso si presenta il problema di dove conservare i dati da visualizzare. Con uno schema analogo a quello di fig. 10.10

sarebbero necessarie quattro porte PIO, per la visualizzazione di 16 cifre, con l'unico scopo di mantenere memorizzati i dati forniti dalla CPU.
Conviene allora ricorrere all'uso delle memorie generalmente predisposte per questa utilizzazione, ad esempio quelle del tipo "dual port"; esse presentano ingressi di controllo e di indirizzamento separati per la scrittura per la lettura, per cui lo schema si riduce a quello indicato in fig. 10.13.

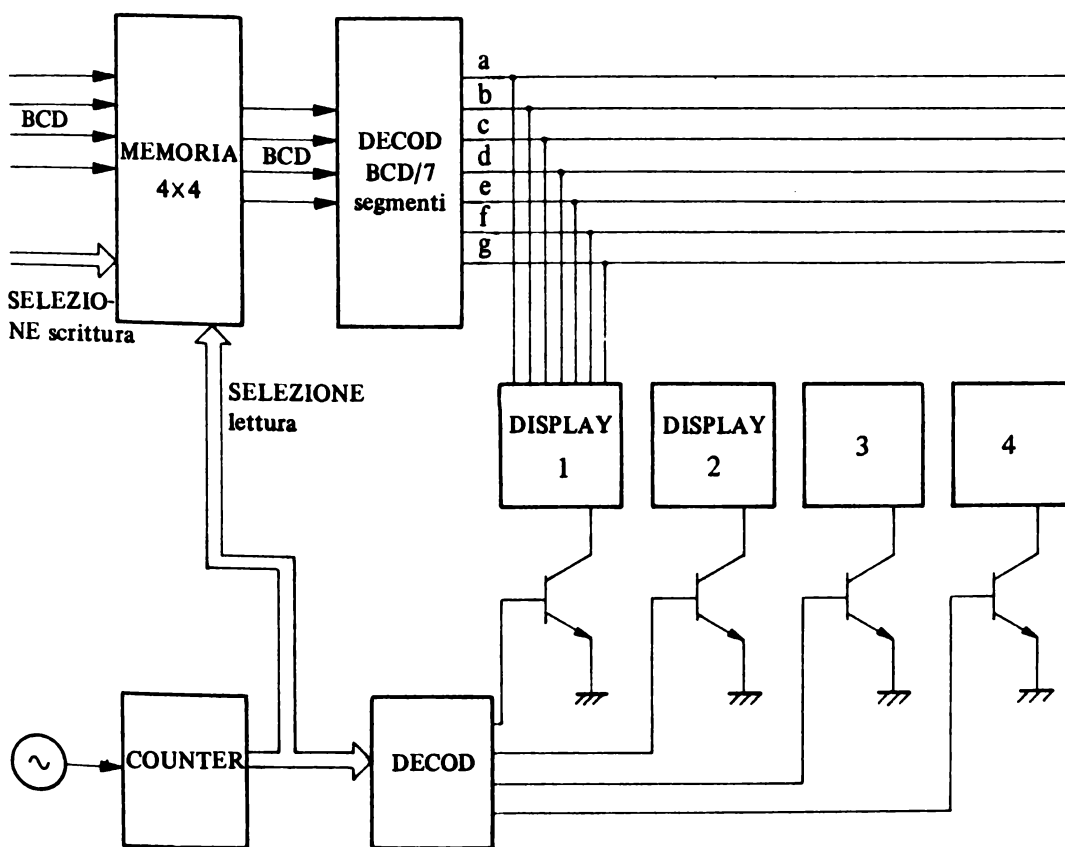



Fig. 10.13
Schema per l'impiego di 4 display a 7 segmenti con l'uso di una memoria.

La memoria può essere ad esempio del tipo SN74170 con la quale è possibile sia la scrittura che la lettura contemporanea. La posizione della cifra nella memoria corrisponde alla posizione sul display per cui si può ottenere una rappresentazione diversa semplicemente cambiando la posizione dei digit in memoria.

10.2.4. *Visualizzatori a matrice*

Quanto finora visto è relativo all'utilizzazione di display a sette segmenti mediante i quali è possibile ottenere la visualizzazione delle 10 cifre ed eventualmente di qualche carattere alfabetico.
Se si vogliono ottenere invece delle informazioni alfanumeriche si deve ricorrere all'uso di display, che funzionano sempre sullo stesso principio di LED, ma presentano un maggiori numero di diodi. Questi ultimi sono organizzati o a matrici di punti con 5 x 7 elementi, oppure a 18 segmenti. Il controllo di questi dispositivi avviene in modo simile a quello visto per i display a 7 segmenti con l'uso del multiplexer.
Si supponga di avere un display a matrice di 5 x 7 punti con il quale si vuole rappresentare la lettera A, come riportato nella fig. 10.14.

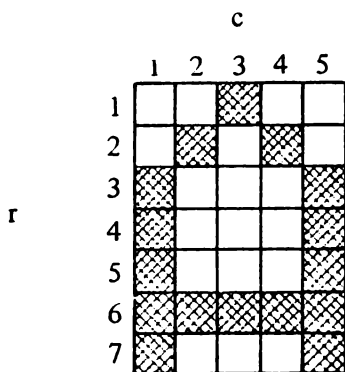



Fig. 10.14.
Rappresentazione della lettera A in un display a matrice 5 x 7.

Se si suppone di organizzare la visualizzazione per colonne si devono avere 7 uscite che vanno ad interessare le 7 righe e una serie di 5 interruttori mediante i quali si fornisce la tensione di alimentazione di una colonna alla volta, come è riportato nella fig. 10.15.
Per la visualizzazione della lettera A si deve inviare il codice di 7 bit 1111100 alla porta A, chiudere l'interruttore I_1 con l'invio del codice 10000 alla porta B. Si invia successivamente il codice 0100010 e si chiude l'interruttore n. 2 con il codice 01000, ecc.

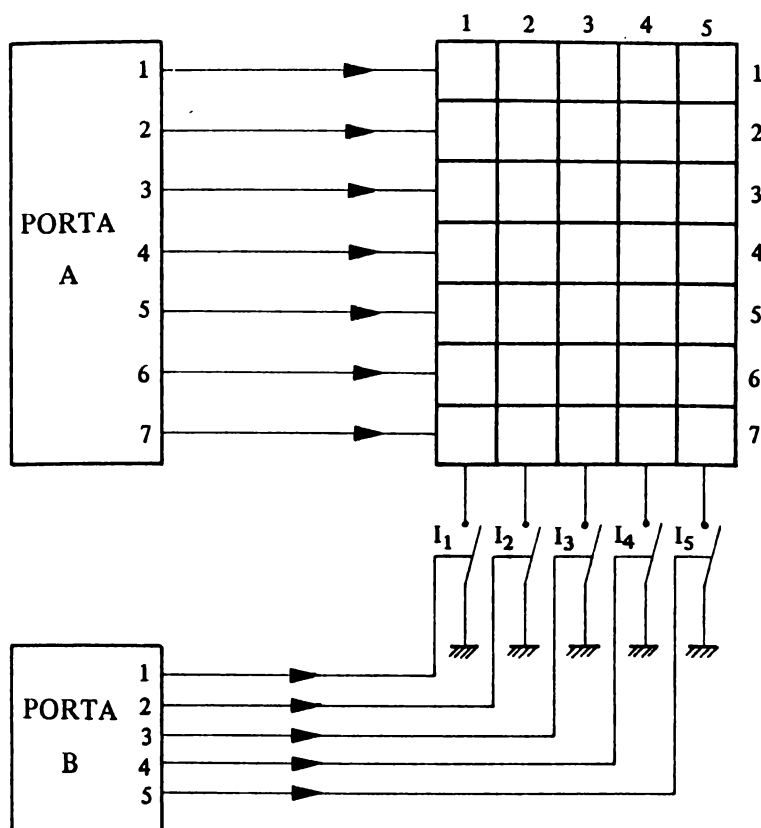



Fig. 10.15
Schema di principio per l'utilizzazione di un display a matrice 7 x 5.

Un circuito che realizza quanto necessario è analogo a quello riportato nella figura 10.13; la memoria deve essere stata opportunamente scritta dal microprocessore in modo da contenere la configurazione dei caratteri che devono essere visualizzati.
Un circuito che presenta una semplice struttura hardware è indicato in fig. 10.16.

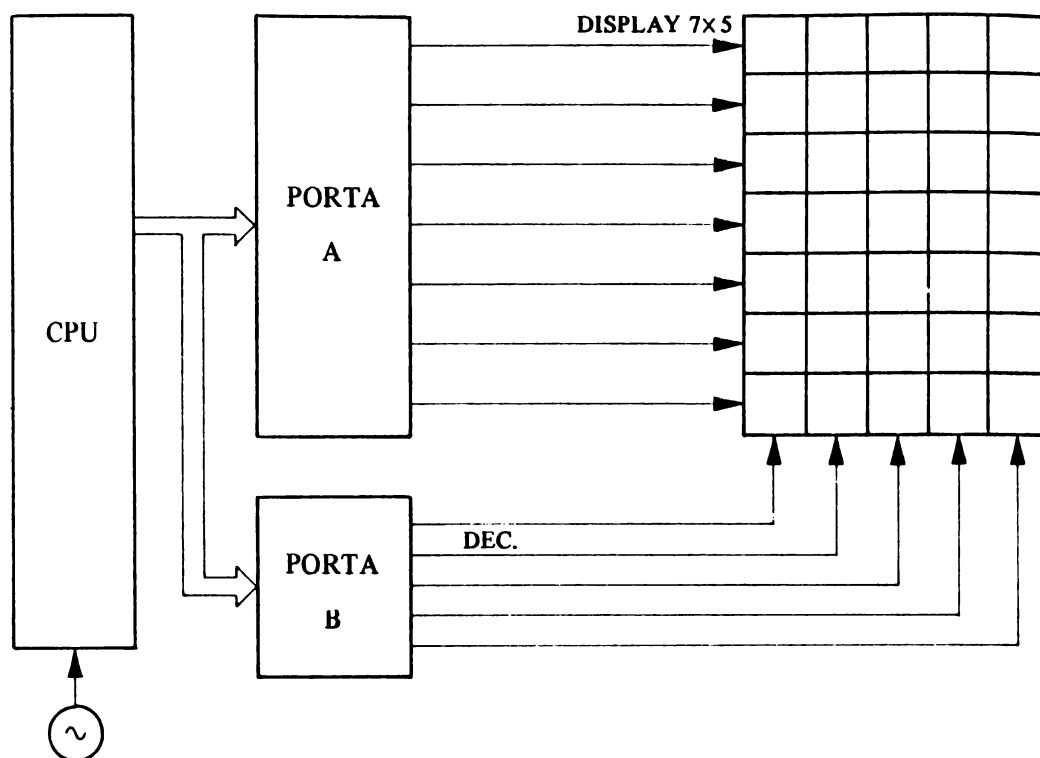



Fig. 10.16
Schema per l'uso di display a matrice 7 x 5.

Il microprocessore riceve una richiesta di interruzione, ad intervalli regolari di tempo; ad ogni interruzione esso invia due configurazioni di bit, una per eccitare una delle colonne del display, ed una che individua la colonna desiderata. Questo modo di procedere, che richiede una struttura hardware particolarmente semplice, ha il difetto di impegnare il microprocessore per il rinfresco del display e per la "traduzione" di ogni carattere nella corrispondente configurazione di bit necessaria per ottenere la visualizzazione desiderata.
Sono disponibili però delle memorie ROM, dette generatori di caratteri, alle quali è sufficiente sia fornito all'ingresso il codice ASCII del carattere perché esse forniscano in uscita i cinque codici a sette bit relativi alle colonne. In tal caso si organizza diversamente l'intera struttura hardware come si può vedere in fig. 10.17.

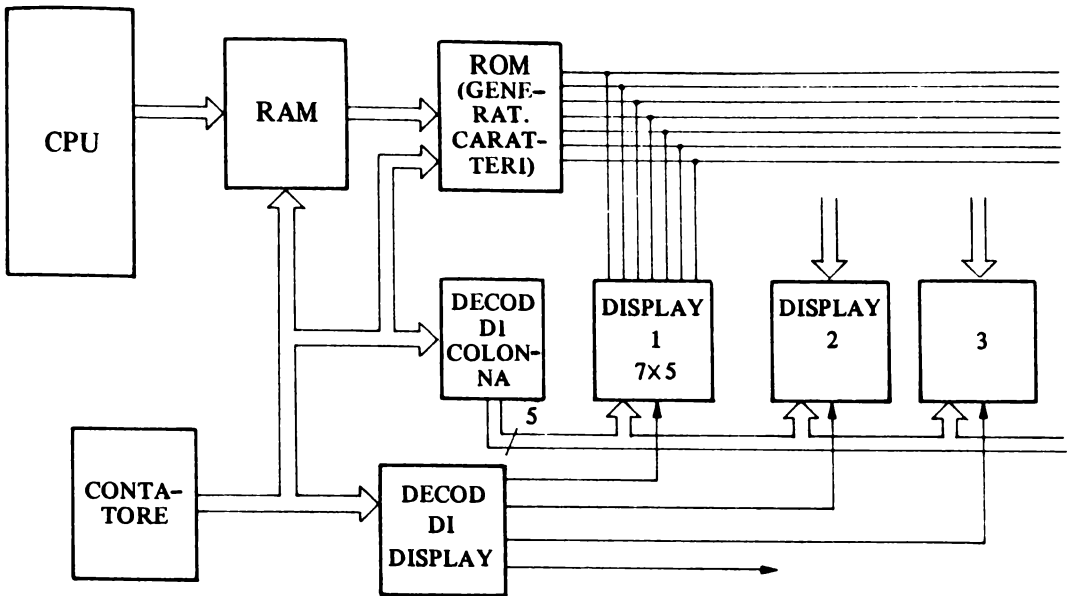



Fig. 10.17
Impiego di un generatore di caratteri per display alfanumerico a LED.

Il microprocessore scrive su una normale memoria RAM la sequenza di caratteri in codice ASCII che si devono visualizzare. Il contenuto di ogni locazione della RAM, selezionata dal contatore, è inviato all'ingresso della memoria ROM generatrice di caratteri.
Il contatore seleziona inoltre nella ROM quella locazione che corrisponde ad una determinata colonna della matrice di diodi, e, tramite il decodificatore, attiva anche la colonna sul display che deve essere visualizzata in quell'intervallo di tempo. Il sistema provvede automaticamente al rinfresco dei vari display e quindi a visualizzare quanto contenuto nella memoria RAM. In questo caso il microprocessore deve solo aggiornare tale contenuto quando desidera cambiare le informazioni da visualizzare: esso quindi vede il display come una normale periferica dove inviare dei dati.

10.3. Tastiere

Le tastiere sono di solito realizzate con un certo numero di interruttori, uno per ogni tasto, disposti secondo una matrice di m righe e di n colonne, in modo tale che l'attivazione di un tasto chiude il collegamento tra una riga ed una colonna, come indicato schematicamente in fig. 10.18.

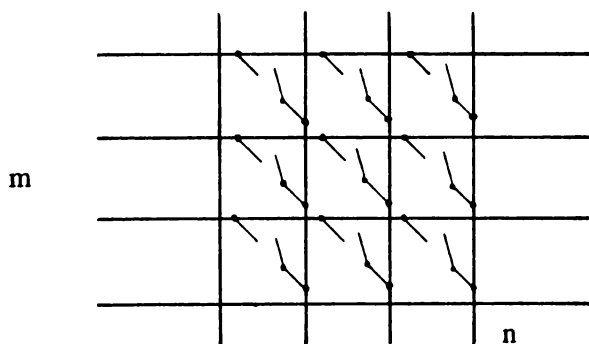



Fig. 10.18
Schema di principio di una tastiera.

Per interfacciare una tastiera così organizzata al microprocessore occorre come al solito utilizzare delle porte di ingresso e di uscita: attraverso queste il microprocessore deve poter individuare in modo univoco quale è il tasto attivato.
E' evidente che necessita un opportuno software al solo scopo di effettuare tale individuazione, e che tale software dipenderà dal modo in cui si è realizzato il circuito di interfaccia.
Un metodo semplice per individuare il tasto attivato è quello che consiste nello scandire le righe della matrice in successione, inviando degli opportuni segnali, e nel verificare se tali segnali si presentano sulle colonne; se ciò avviene significa che almeno uno dei tasti è stato attivato.
Un metodo molto semplice per individuare il tasto attivato è quello che consiste nello scandire le righe della matrice in successione, inviando degli opportuni segnali, e nel verificare se tali segnali si presentano sulle colonne; se ciò avviene significa che almeno uno dei tasti è stato attivato.
se è attivato un tasto, ciò fa sì che si porti allo stesso livello anche la colonna corrispondente a quel tasto, quando è attiva la riga cui questo è collegato.
Il software di scansione della tastiera, al solo scopo della individuazione dei tasti attivati, deve allora inviare nella porta di uscita la configurazione 0001, e tale configurazione deve permanere per un intervallo di tempo τ.

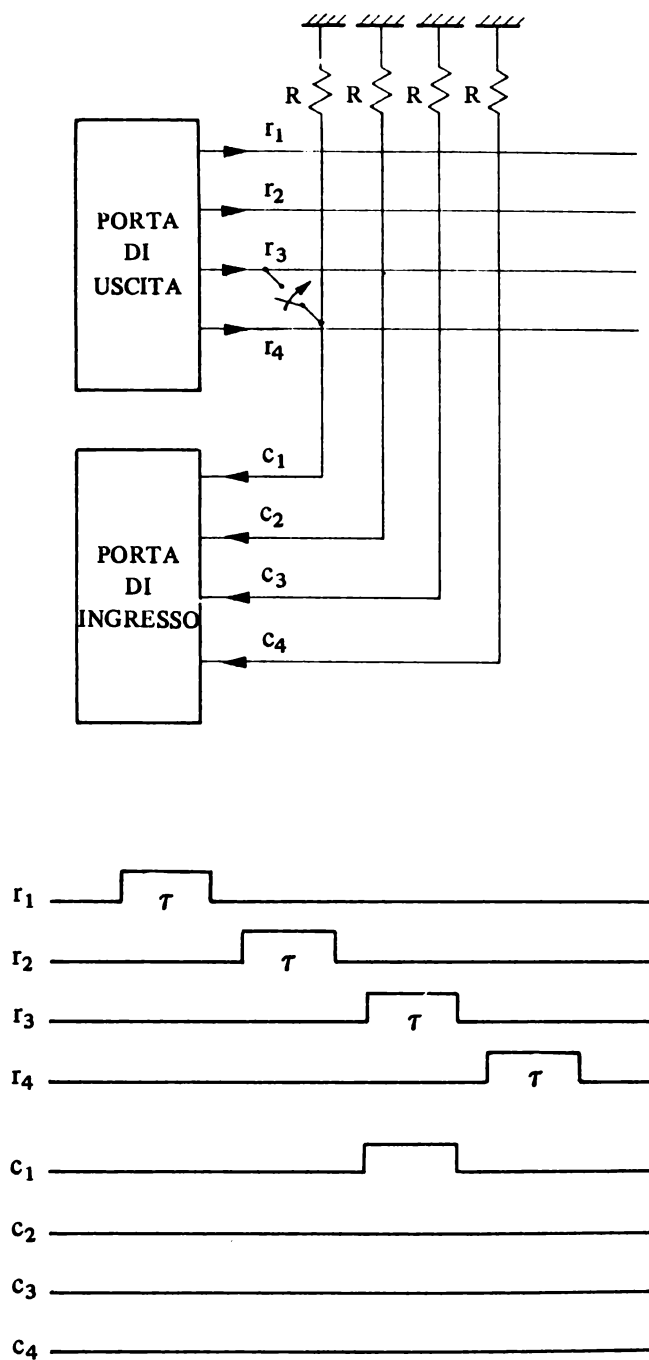



Fig. 10.19
Scansione su una tastiera (a) e diagramma temporale dei segnali di uscita ed ingresso (b).

Successivamente il microprocessore effettua una lettura della porta di ingresso per individuare se è stato attivato qualcuno dei tasti collegati alla prima riga. Effettuata la lettura, e l'eventuale memorizzazione dello stato delle colonne, il microprocessore procede con l'invio alla porta di uscita della configurazione che permette di scandire la seconda riga ,0010, quindi procede ad una nuova lettura dello stato delle colonne e così via ciclicamente.
A parte il frequente impiego del microprocessore, tale organizzazione presenta un inconveniente legato alle caratteristiche meccaniche dei tasti, cioè degli interruttori impiegati: infatti il microprocessore esegue la lettura dello stato delle colonne in istanti predeterminati (ad esempio a metà di un intervallo di tempo τ) e può succedere, a causa di rimbalzi del contatto di un interruttore, che non rilevi la presenza del livello logico alto sulla colonna corrispondente, per cui non può individuare con sicurezza se il tasto è stato attivato. Occorre quindi simulare via software un circuito antirimbalzo e ciò provoca un ulteriore appesantimento dei programmi di gestione.
Si deve aggiungere a ciò che una volta che un tasto sia stato attivato e che il microprocessore lo abbia rilevato, il tasto stesso deve essere considerato nuovamente attivato solo dopo che sia stato rilasciato e successivamente ancora premuto dato che la scansione e rilevazione si attua in genere con tempi dell'ordine dei 10 ÷ 20 ms, abbondantemente sufficienti perché i rimbalzi si esauriscano.
Un'altro tipo di inconveniente si presenta quando si premono più tasti contemporaneamente; anche in questo caso si possono adottare diverse soluzioni.
Con un hardware opportuno si può far sì che quando un tasto è premuto la pressione di altri non alteri lo stato delle colonne di uscita che si leggono con il microprocessore: perché si possa avere un cambiamento del contenuto delle colonne di uscita è necessario che sia rilasciato il tasto precedentemente premuto.
Nella matrice indicata in fig. 10.20 si immagini invece siano chiusi contemporaneamente i tasti (r_1, c_3), (r_2, c_3) e (r_2, c_2) e sia attiva la riga r_1: in questo caso la tensione sulla colonna c_3 è a livello logico alto come ci si aspetta, ma è a tale livello anche quella della riga r_2: infatti la tensione di scansione si propaga attraverso i 3 tasti e quindi anche la colonna c_2 si porta a livello alto: la conclusione è che appare attivato anche il tasto (r_1, c_2).
Per ovviare a tali convenienti occorre inserire dei diodi di blocco, come in fig. 10.21, in serie ad ogni interruttore: ora la chiusura contemporanea dei tre tasti come nel caso precedente fa sì che solo c_3 assuma il valore logico alto quando è attiva la riga di scansione r_1.

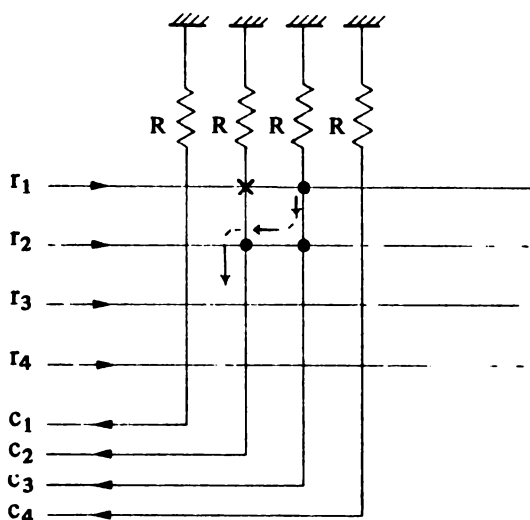



Fig. 10.20
Tastiera con tre tasti premuti.

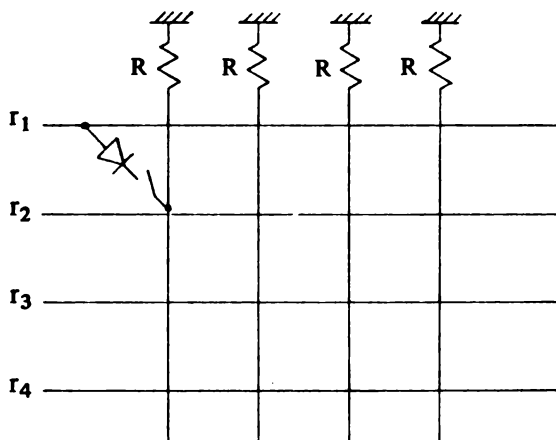



Fig. 10.21
Uso dei diodi in una tastiera.

Un altro problema che si presenta nella gestione di tastiere è quello relativo al tempo impiegato dal microprocessore per individuare i tasti premuti. Se non si sono predisposte altre soluzioni il microprocessore deve procedere ad una continua scansione per cui esso è completamente impegnato solo per controllare lo stato della tastiera e ciò può non essere accettabile.

Una soluzione che si può adottare per evitare questo inconveniente è riportata nella fig. 10.22.

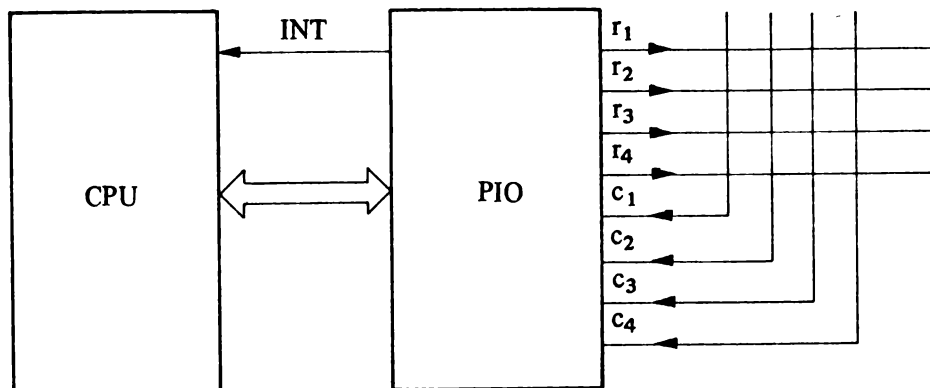



Fig. 10.22
Impiego di una PIO nella scansione di una tastiera.

La PIO deve essere programmata a funzionare con una delle sue porte in modo 3, con quattro bit di ingresso e quattro di uscita, con la possibilità inoltre di richiedere una interruzione quando una delle colonne assume un valore logico alto. In questo caso una completa scansione della tastiera richiede quattro istruzioni di uscita oltre alle quattro di ingresso.
Un altro modo di procedere è il seguente: invece di portare una riga alla volta allo stato alto, si portano tutte le righe in tale stato e si leggono i valori assunti dalle colonne. Se un tasto è attivato una delle colonne assumerà il livello logico alto, ma da ciò non è ancora possibile dedurre quale sia il tasto premuto in quanto non si ha alcuna indicazione della relativa riga. Si procede allora ad una riprogrammazione della PIO che cambi il verso di utilizzazione dei vari bit: quelli che erano di ingresso diventano di uscita e viceversa; successivamente si invia un livello alto su tutte le colonne e si leggono le righe. Ora è possibile avere l'informazione di quale è stato il tasto premuto; si tratta solamente di decodificare le due informazioni ricevute dall'esterno. Si noti che in questo caso sono necessarie solo due operazioni di scrittura e due di lettura invece di otto come nel caso precedente. Si deve però cambiare la modalità di funzionamento della porta interessata e ciò comporta ulteriori istruzioni: nel caso di una porta PIO sono necessarie per questa operazione solamente due istruzioni e

quindi questa tecnica è preferibile rispetto a quella più intuitiva illustrata per prima. Un vantaggio maggiore si ha se la tastiera è da 64 tasti (8x8): le due porte A e B della PIO sono usate allora alternativamente di ingresso e di uscita.

10.4. Tastiere codificate

Sono disponibili sul mercato delle tastiere le quali, non appena è premuto un tasto presentano su un certo numero di linee di uscita, che sono collegate ad una porta di ingresso del microprocessore, la configurazione che individua, in un certo codice, il tasto attivato. Inoltre esse in una ulteriore linea di uscita generano un impulso che avvisa dell'avvenuta attivazione di un tasto e che può essere utilizzato per richieste di interruzione. Tali tipi di tastiere sono ancora organizzati a matrice ed inoltre hanno già predisposti dei circuiti che effettuano la scansione della tastiera e che codificano le linee di ritorno secondo un codice prefissato.
In fig. 10.23 è indicato uno schema a blocchi che realizza una tastiera codificata di (mxn) tasti.
Come si vede è presente un generatore di clock il quale eccita un contatore binario, di modulo pari al numero di colonne presenti nella tastiera, le cui uscite vanno a comandare un decodificatore ad n uscite, attive a livello logico basso. In tal modo ad ogni impulso di clock è aggiornato il contenuto del contatore binario e quindi è attivata una sola delle uscite dal decodificatore. Le righe della matrice di tasti sono collegate a dei circuiti antirimbalzo le uscite dei quali vanno ad eccitare un codificatore che presenta alcune particolarità. Innanzitutto le sue uscite contengono il valore, in binario, dell'ingresso che in quell'istante è al valore logico basso. Nel caso più ingressi abbiano tale valore il codificatore genera un codice corrispondente a quello che identifica la linea, a livello logico basso, di numero d'ordine più elevato. Oltre a queste uscite il codificatore ne presenta un'altra, DV, la quale è attiva solo se almeno una delle linee di ingresso del codificatore è al livello logico basso. Il buffer di uscita è di tipo latch e memorizza sia l'uscita del contatore sia quella del decodificatore non appena è attiva l'uscita DV. Quest'ultima, quando viene attivata, setta anche un flip-flop la cui uscita assume in significato di richiesta di interruzione.
Il funzionamento del circuito è il seguente: non appena è premuto un tasto è inviato al codificatore un valor logico zero sulla linea corrispondente a quel tasto solo se è a valor logico zero anche la colonna di scansione cui è collegato il tasto stesso.

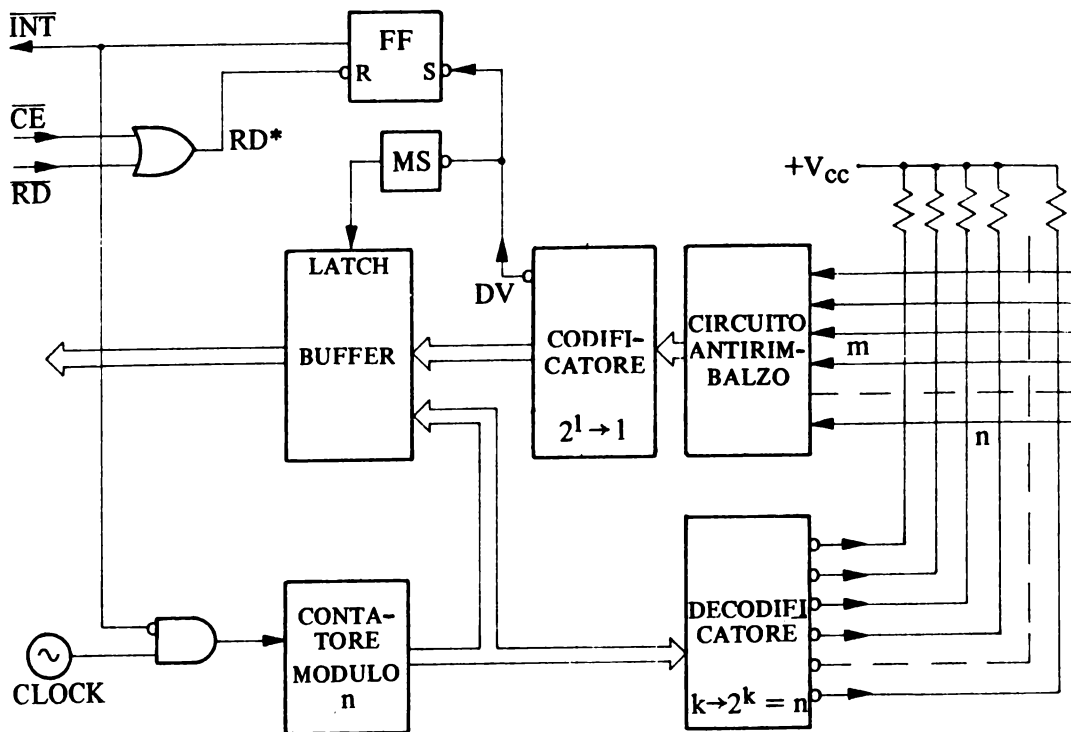



Fig. 10.23

Ciò provoca l'attivazione dell'uscita DV la quale setta il FF di richiesta di interruzione, e quindi blocca il contatore, oltre a far si che venga memorizzato nel buffer di uscita (come numero di riga e di colonna) il codice corrispondente al tasto premuto.
Da questo momento in poi il contatore rimane bloccato fino a quando il microprocessore non effettua la lettura del buffer tramite il segnale RD*, il quale resetta il FF di richiesta di interruzione e quindi permette il proseguimento della scansione della matrice.
Lo schema illustrato non permette la gestione simultanea di più tasti; inoltre il microprocessore deve poter effettuare la lettura del buffer in tempi

brevi in quanto l'operatore non ha alcuna segnalazione che il tasto attivato sia stato correttamente letto da parte dell'unità centrale.
Nello schema indicato in fig. 10.23 evidentemente ogni tasto è individuato dalla riga e dalla colonna alle quali esso è collegato. Ad esempio, se la tastiera ha una disposizione di tasti del tipo di una macchina da scrivere, il tasto corrispondente alla lettera A, se si vuole che la tastiera generi delle codifiche ASCII a 7 bit, deve essere collegato tra una riga ed una colonna in modo tale da dare in uscita la configurazione 41H cioè 1000001. Se gli ultimi quattro bit sono quelli che identificano il numero d'ordine della colonna di scansione, e se i tre bit più significativi sono relativi al numero d'ordine della linea di ritorno, allora il tasto dovrà effettuare il collegamento tra la seconda colonna e la quinta riga (la numerazione delle righe e delle colonne parte da zero). E' chiaro allora che la disposizione (layout) dei tasti deve essere organizzata in modo opportuno così da formare i codici desiderati per ogni tasto.
Da quanto detto se si vuole realizzare una tastiera codificata secondo la disposizione tipica di una macchina da scrivere, con codifica ASCII, ne risulta che i collegamenti ai vari tasti possono risultare alquanto complessi. Per evitare ciò tra il contatore ed il decodificatore ed il buffer di uscita alcune volte è interposto un ulteriore codificatore che effettua la conversione tra il codice riga-colonna della matrice di tasti, realizzata con una disposizione la più semplice possibile per rendere facile la costruzione, ed il corrispondente codice ASCII; tale codificatore è usualmente realizzato con una ROM.
Riassumendo i problemi che si presentano nella gestione di tastiere, codificate o meno, possono essere risolti sia con una opportuna organizzazione dell'hardware sia con del software più o meno complicato.
Fortunatamente però sono stati realizzati da varie case costruttrici di microprocessori dei dispositivi programmabili che permettono una gestione semplice ed efficace sia di tastiere che di display.

10.5. Dispositivi programmabili per la gestione di tastiere e display

A tale classe di dispositivi appartiene l'8279 il quale realizza contemporaneamente le interfaccie, fra una tastiera e un sistema di visualizzazione, con il microprocessore.
Con l'8279 è possibile collegare al microprocessore, in modo molto semplice, una tastiera costituita da un numero di tasti che può arrivare ad un massimo di 64 e gestire direttamente un visualizzatore composto da, al massimo, 16 caratteri a sette segmenti con punto decimale.
In fig. 10.24 sono indicati i diversi segnali relativi al funzionamento dell'8279: a sinistra sono riportati quelli utilizzati per lo scambio di informa-

PIN NAMES

DB_{0-7}	I/O	DATA BUS (BI-DIRECTIONAL)
CLK	I	CLOCK INPUT
RESET	I	RESET INPUT
$\overline{CS}$	I	CHIP SELECT
$\overline{RD}$	I	READ INPUT
$\overline{WR}$	I	WRITE INPUT
A_0	I	BUFFER ADDRESS
IRQ	O	INTERRUPT REQUEST OUTPUT
SL_{0-3}	O	SCAN LINES
RL_{0-7}	I	RETURN LINES
SHIFT	I	SHIFT INPUT
CNTL/STB	I	CONTROL/STROBE INPUT
OUT A_{0-3}	O	DISPLAY (A) OUTPUTS
OUT B_{0-3}	O	DISPLAY (B) OUTPUTS
$\overline{BD}$	O	BLANK DISPLAY OUTPUT

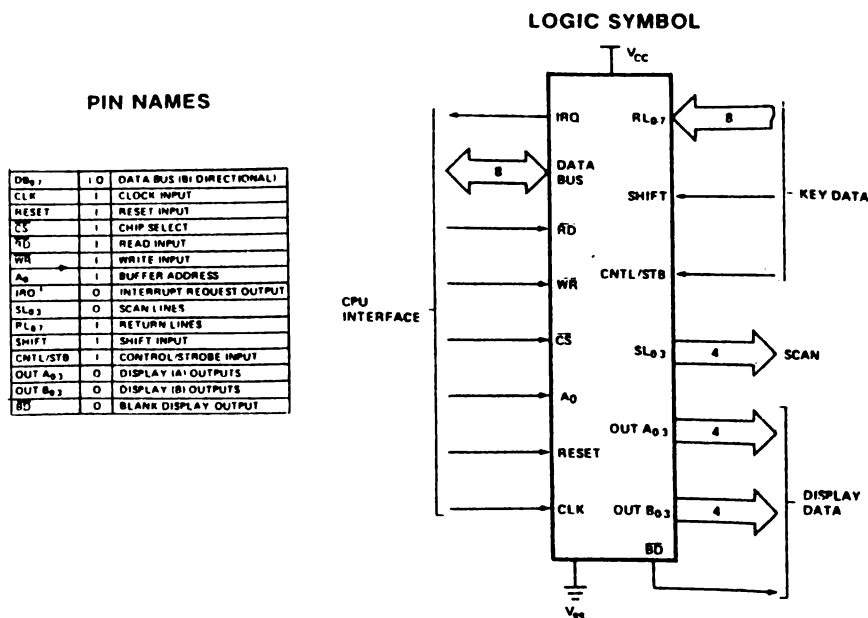



Fig. 10.24
I segnali nell'8279. (Intel, Component Data. Catalog 1979).

zioni con l'unità centrale; a destra quelli che consentono la gestione della tastiera e del display.
Per quanto riguarda i primi, sono presenti un bus di dati, attraverso il quale il microprocessore può inviare all'8279 i comandi di programmazione o i dati da visualizzare, e ricevere i codici provenienti dalla tastiera oltre alle informazioni relative allo stato in cui si trova in quel momento l'8279 stesso.
Gli ingressi di controllo CS/, WR/, RD/ consentono rispettivamente di selezionare il componente e di individuare in modo univoco l'operazione di scrittura o di lettura. E' inoltre presente un ingresso A0 tramite il quale il microprocessore può informare l'8279 se il byte presente nel bus dei dati deve essere interpretato come un comando oppure come dato effettivo.
I segnali RESET e CLOCK servono rispettivamente per inizializzare il dispositivo in un ben determinato modo di funzionamento e per stabilire la frequenza di scansione della tastiera e di multiplexing del visualizzatore. L'8279 può richiedere interruzioni all'unità centrale tramite il segnale IRQ.
Per quanto riguarda invece i segnali relativi alla tastiera ed al visualizzatore, le linee SLO-3 servono contemporaneamente per scandire la tastiera e per selezionare i visualizzatori.

Oltre alle 8 linee di ritorno dalla matrice dei tasti, RLO-7, sono presenti le linee di SHIFT e di CNT/STB le quali consentono, a seconda del loro valore logico, di associare ben quattro significati diversi per ogni tasto attivato. Nelle linee OUT A0-3 e OUT B0-3 l'8279 invia i comandi per l'attivazione dei segmenti del singolo visualizzatore; con il segnale BD/ invece esso ne può inibire o meno il funzionamento.
Lo schema a blocchi del dispositivo è riportato nella fig. 10.25.

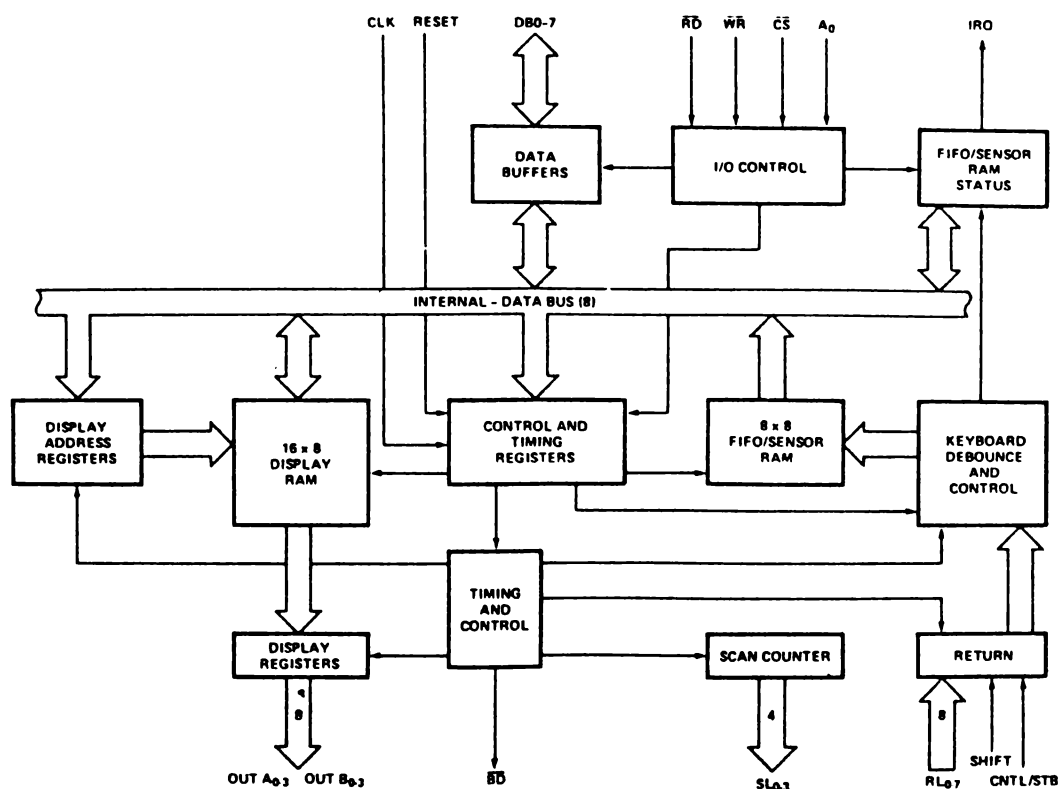



Fig. 10.25
Schema a blocchi dell'8279. (Intel, Component Data. Catalog 1979).

Si può notare in esso la presenza di una memoria RAM da 16 byte nella quale l'8279 memorizza le informazioni, relative ai caratteri da visualizzare, inviate dalla CPU. Una ulteriore memoria RAM, con caratteristiche FIFO, conserva invece le informazioni relative ai tasti attivati sulla tastiera. Queste ultime sono acquisite solo dopo un controllo effettuato dall'8279, per eliminare i segnali spuri di rimbalzo dei tasti.

10.5.1. *Caratteristiche di funzionamento dell'8279*

Per quanto riguarda la gestione della tastiera il dispositivo può essere programmato per funzionare in tre modi distinti.
Nel modo di funzionamento a "tastiera scandita" l'8279 attiva ciclicamente e con frequenza fissa le linee di scansione SLO-3, così che in ogni istante una sola di esse risulta attiva.
Lo schema di applicazione di tale modo è riportato nella fig. 10.26, dove è indicata una tastiera a matrice con 4 colonne di scansione, attivate secondo il diagramma temporale ivi indicato, e 8 righe di ritorno.

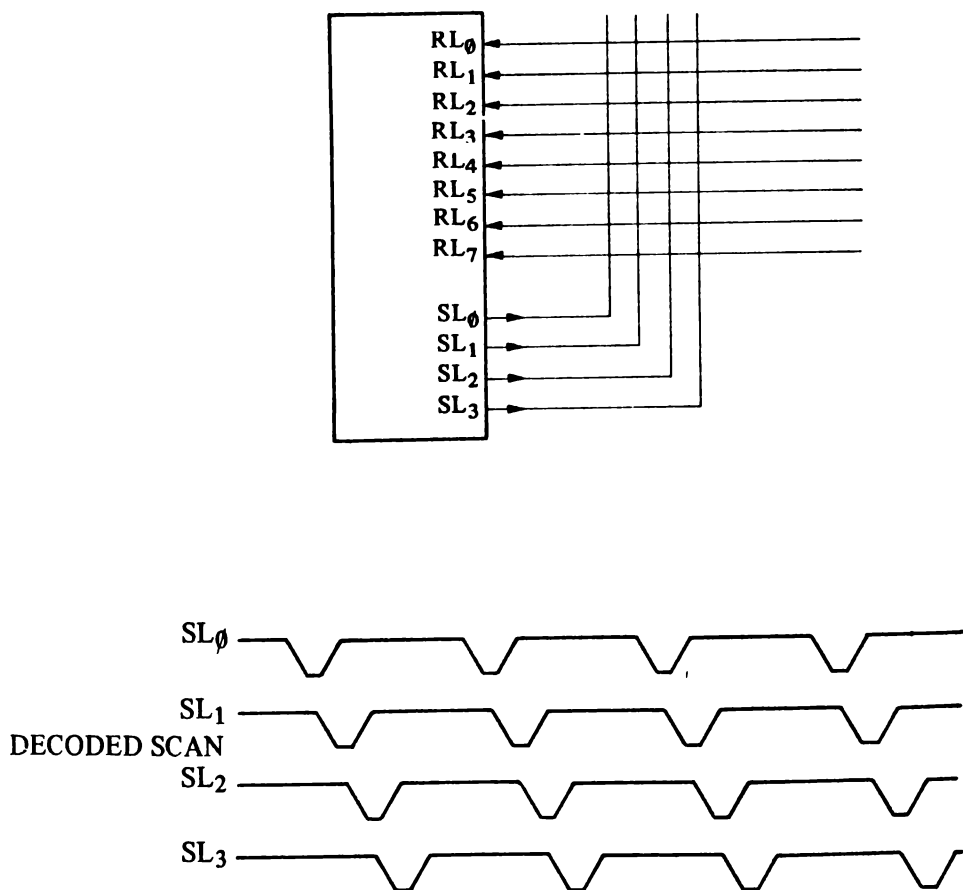



Fig. 10.26
Modo di funzionamento a tastiera scandita decodificata.
(Intel, Component Data. Catalog 1979).

Durante l'intervallo di tempo in cui una colonna SL è a valore logico basso, l'8279 analizza lo stato delle otto linee di ritorno: se è stato premuto un tasto collegato a quella colonna di scansione, anche una delle linee di ritorno risulterà bassa. Tale situazione è rilevata dall'8279, il quale memorizza in un registro temporaneo sia il numero della riga che quello della colonna di scansione attive in quel momento, cioè le coordinate del tasto premuto.
Per eliminare i segnali spuri dovuti ai rimbalzi dei tasti l'8279 attende una scansione completa della tastiera e quindi va a verificare se si ripresentano le stesse condizioni.
Se ciò accade l'informazione presente nel registro temporaneo è trasferita nella memoria FIFO, ritenendo quindi effettiva l'attivazione di quel tasto.
Occorre aggiungere che durante l'analisi delle linee di ritorno l'8279 memorizza nel registro temporaneo anche il valore logico presente ai due ingressi SHIFT e CNTL e quindi anche queste informazioni sono memorizzate nella FIFO.
Non appena l'8279 ha effettuato una scrittura nella FIFO, esso attiva il segnale di uscira IRQ per la richiesta di interruzioni: in tal modo si avvisa il microprocessore che è stato premuto un tasto e che il relativo codice si trova memorizzato nella FIFO stessa. Tale richiesta di interruzione cesserà non appena il microprocessore avrà letto la FIFO per prelevare il codice relativo al tasto premuto. Nel caso siano state attivati e accettati in successione più tasti, la richiesta di interruzione permarrà attiva fintanto che il microprocessore non avrà letto tutte le informazioni contenute nella FIFO. In altre parole la richiesta di interruzione resta attiva per indicare che c'è ancora da leggere qualche codice di tasto. Dato che il tempo necessario alla scansione completa della tastiera è dell'ordine della decina di ms, è evidente che il microprocessore ha tutto il tempo sufficiente per leggere la FIFO, purché le interruzioni non siano disabilitate e non accada che pervenga una richiesta a più elevata priorità di quella della tastiera.
Per quanto riguarda la gestione di quest'ultima nel modo a scansione, occorre stabilire, per il tramite di comandi di programmazione, come si deve comportare l'8279 quando sono premuti più tasti "contemporaneamente" inteso qui nel senso che la pressione di più tasti avviene entro lo stesso intervallo di tempo impiegato per una scansione completa. I criteri utilizzati sono essenzialmente due; col primo, che prende il nome di "2-key lockout", si impone che non possano essere accettati tasti attivati contemporaneamente.
Nel secondo criterio di scansione, detto "N-key rollover", l'8279 accetta tutti i codici dei tasti premuti contemporaneamente; non è in grado però di discriminare l'ordine temporale con cui sono stati premuti e la memorizzazione dei relativi codici avviene secondo l'ordine con cui la scansione della tastiera li rivela.

Per quanto riguarda il codice associato a ciascun tasto, e memorizzato nella FIFO, esso è composto da 8 bit secondo lo schema di fig. 10.27.

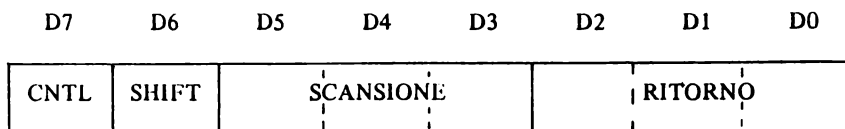



Fig. 10.27
Codice relativo all'identificazione di un tasto come memorizzato nella FIFO.

I tre bit meno significativi individuano il numero d'ordine della linea di ritorno corrispondente a quel tasto; i successivi tre bit individuano invece il numero d'ordine della colonna di scansione cui quel tasto è connesso; i rimanenti due bit, i più significativi, assumono il valor logico degli ingressi SHIFT e CNTL rispettivamente.
Se, ad esempio, è attivato il tasto relativo alla colonna numero 5 e alla riga numero 3, il codice memorizzato nella FIFO ha il valore XX10 1011, dove XX indicano i valori assunti dalle due linee di ingresso SHIFT e CNTL.
Nella fig. 10.26 le linee SLO-3 sono utilizzate direttamente per la scansione della tastiera: si dice che questo modo di operare è del tipo decodificato. E' anche possibile impore all'8279 di inviare alle linee SLO-2 un codice numerico che individua le colonne: in tal caso le linee di scansione assumono ciclicamente i valori da 0 a 7, che indicano il numero d'ordine di quale delle otto colonne di scansione in quel momento deve essere attivata.
Per l'utilizzazione di queste informazioni è necessario l'impiego di un decodificatore da tre a otto, secondo lo schema di fig. 10.28.
Come si può notare è ora possibile gestire una tastiera di dimensioni 8x8; se però si utilizzano anche le linee di CNTL e SHIFT è possibile attribuire ad ogni tasto fino a quattro significati diversi. Si può in tal modo gestire con facilità, ad esempio, la tastiera di una telescrivente dove i singoli tasti possono rappresentare sia un carattere minuscolo che uno maiuscolo se è attivato lo SHIFT, ed inoltre la pressione del tasto CONTROL contemporanea a quella di altri tasti fa assumere a questi ultimi particolari significati come ad esempio tabulazione orizzontale, ecc.
Il secondo modo di gestione di una tastiera è quello detto a "matrice di sensori"; in questo caso quella che era prima gestita come una RAM FIFO diventa ora una RAM normale in cui è memorizzata l'immagine, aggiornata, scansione per scansione, dello stato degli interruttori presenti nella matrice. Questo modo di funzionamento è utilizzato in genere per la lettura di segnali provenienti da trasduttori, segnali che devono essere op-

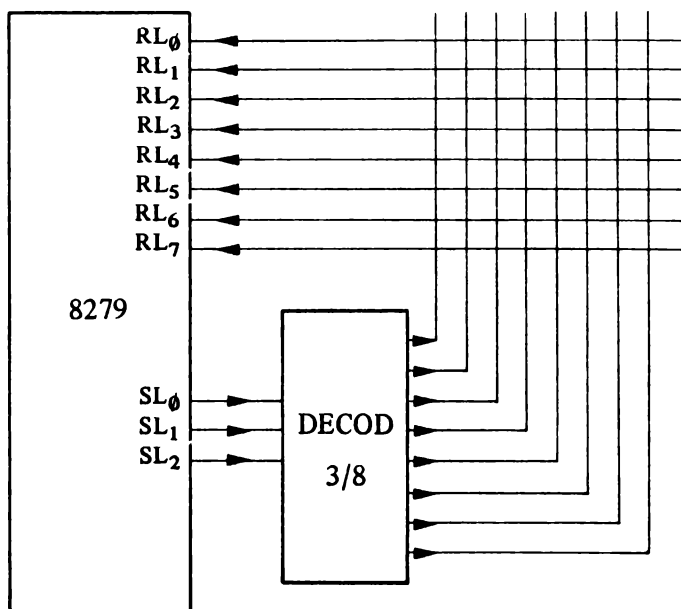



Fig. 10.28
Modo di funzionamento a tastiera scandita codificata.

portunamente condizionati dato che in questo modo di funzionamento il circuito per la prevenzione dei rimbalzi non è attivo. Di solito le connessioni tra le colonne di scansione e le linee di ritorno sono effettuate per mezzo di interruttori elettronici come ad esempio transistor, fototransistor, ecc.

Quando, alla fine di una scansione, l'8279 si accorge che lo stato di un'interruttore è diverso da quello rilevato nella scansione precedente, esso genera una richiesta di interruzione per dar modo alla CPU, verificando il contenuto della RAM, di individuare il o gli interruttori che hanno cambiato di stato. Questo modo di funzionamento ha il vantaggio, rispetto al precedente, che la CPU ha la possibilità di valutare, via software, per quanto tempo un interruttore permane in un certo stato.

Infine nel terzo modo di funzionamento, "strobed input mode", è compito della tastiera inviare il codice di un tasto e attivare contemporaneamente l'ingresso di strobe; ciò fa si che quel codice sia memorizzato nella FIFO con le stesse modalità del modo a tastiera scandita.

10.5.2. *Programmazione dell'8279*

L'8279 contiene nel suo interno 8 registri di scrittura nei quali la CPU invia le informazioni atte a stabilire le modalità di funzionamento. E' presente inoltre un registro di lettura che permette alla CPU di ottenere informazioni relative allo stato in cui si trova in quel momento l'8279.
La scrittura e lettura dei registri si ha solo se l'ingresso A0 è posto a livello logico alto: questo sta ad indicare infatti che il byte trasmesso o ricevuto dalla CPU costituisce rispettivamente un comando od uno stato.
Per l'indirizzamento dei registri di scrittura si utilizzano i tre bit più significati del comando stesso; l'8279 effettua infatti automaticamente la decodifica di tale bit e memorizza quindi sul registro così individuato l'informazione contenuta nei restanti cinque bit meno significativi.
L'8279 è organizzato in modo tale per cui qualsiasi operazione di lettura (ad esempio della FIFO) o di scrittura (ad esempio della RAM del display) deve essere stata preceduta da un comando, inviato dalla CPU, il quale indica all'8279 l'origine o la destinazione del successivo dato trasmesso e le modalità con cui questo dato deve essere interpretato.
I registri di programmazione si possono suddividere in due gruppi: al primo gruppo cui appartengono i registri 0 e 1 i quali servono per programmare il modo di funzionamento della tastiera, il modo di funzionamento del display e la frequenza di scansione.
Nel secondo gruppo di registri devono essere inviate le informazioni atte a qualificare le successive operazioni di scrittura o di lettura di dati secondo quanto detto precedentemente.
Per quanto riguarda il controllo del display, l'8279 è particolarmente adatto per l'uso di dispositivi a 7 segmenti.
I segnali $SL_{0\text{-}3}$, che, come si è visto, sono utilizzati per la scansione della tastiera, servono anche per individuare quale display, nella serie di visualizzatori, è quello che deve essere attivato durante quell'intervallo di scansione. L'informazione relativa alla cifra, o al carattere, da visualizzare, è fornita dal codice, presente sulle line $A_{0\text{-}3}$ e $B_{0\text{-}3}$, che proviene dalle RAM di display interne all'8279 stesso.
In pratica l'8279 legge la RAM suddetta e ne porta all'esterno il contenuto, tramite $A_{0\text{-}3}$ e $B_{0\text{-}3}$, attivando contemporaneamente i segnali $SL_{0\text{-}3}$ che corrispondono all'indirizzo della locazione di RAM letta.
Lo schema che si può utilizzare per il comando del display, nel caso sia formato da un numero di visualizzatori non superiori a 16, è quello indicato in fig. 10.29.

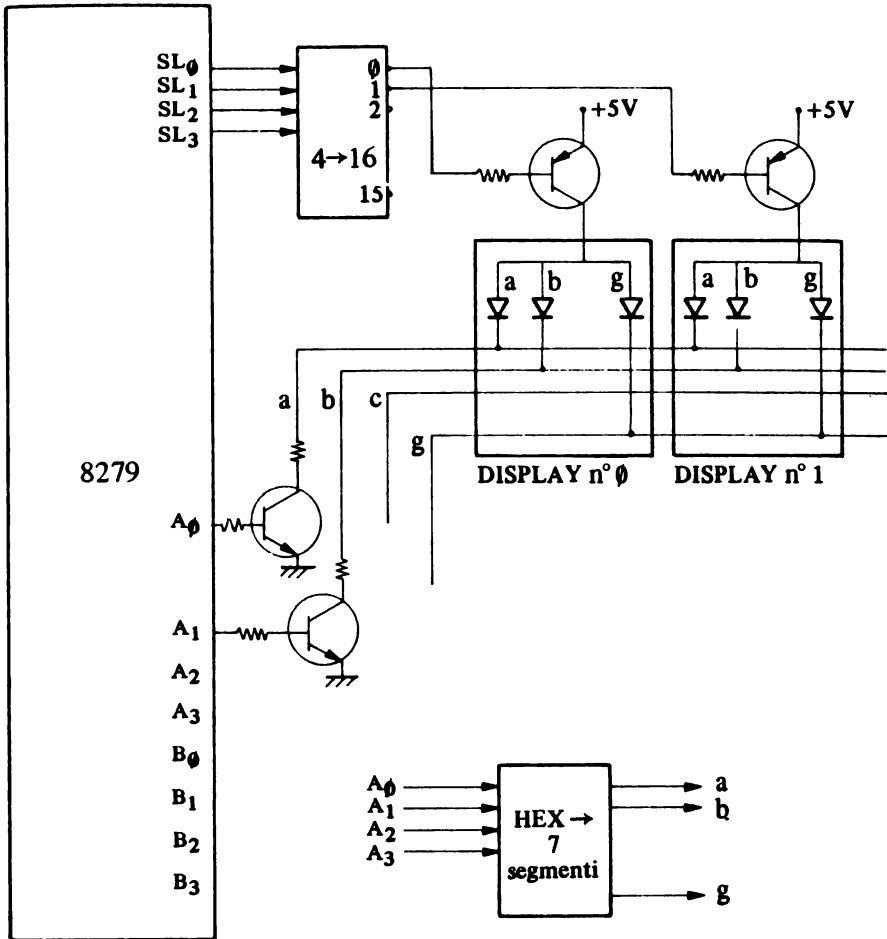



Fig. 10.29
Applicazione dell'8279.

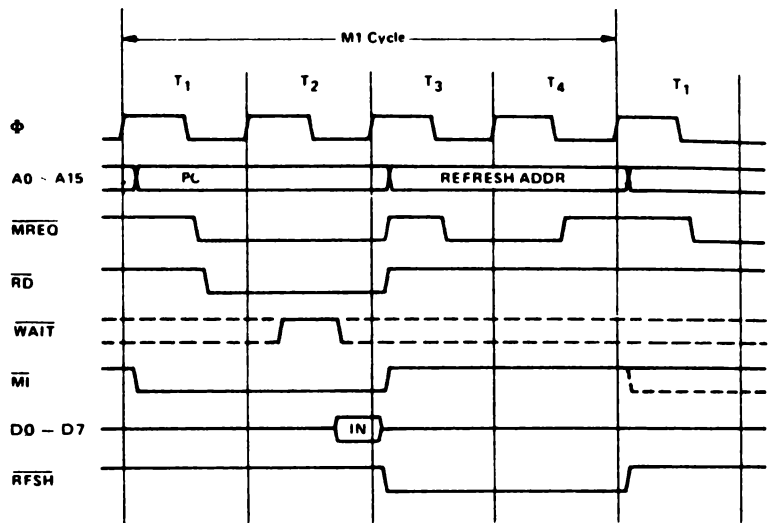
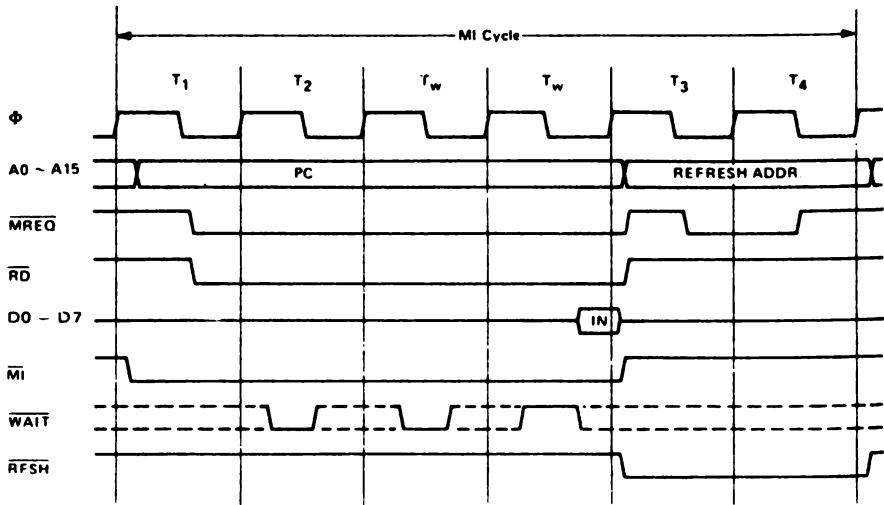
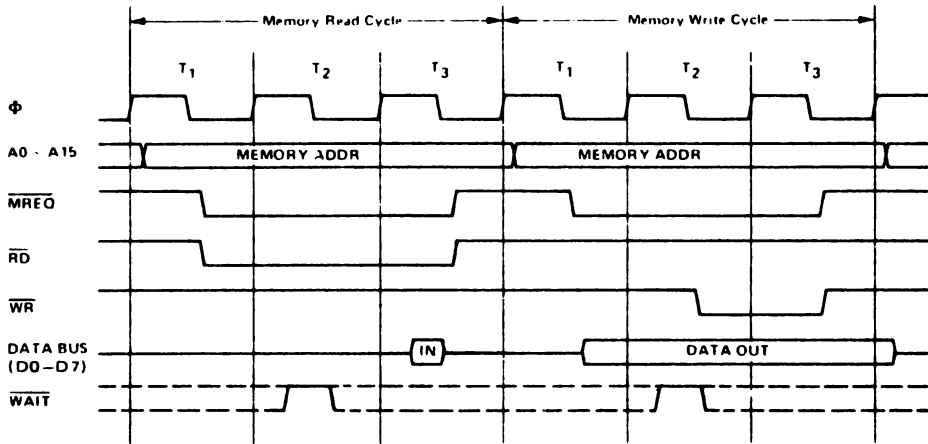
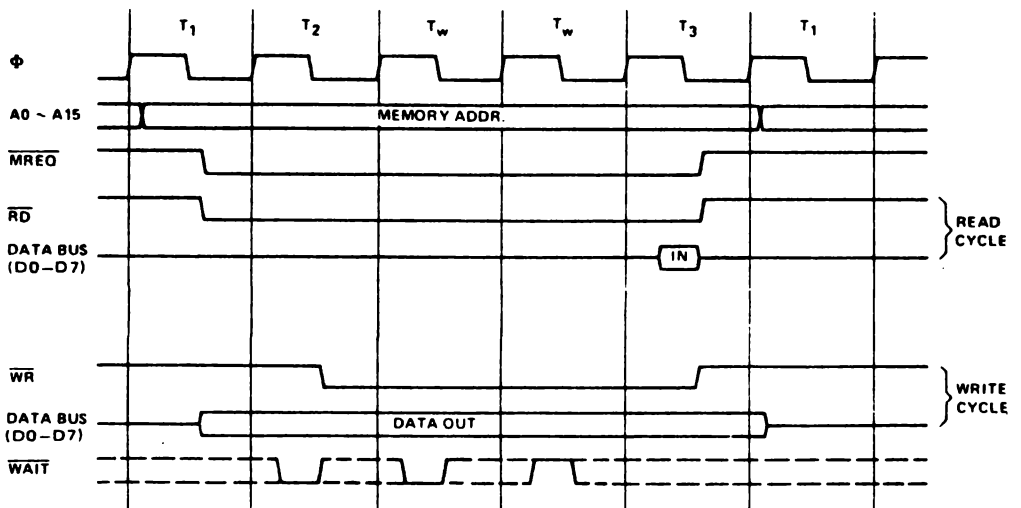
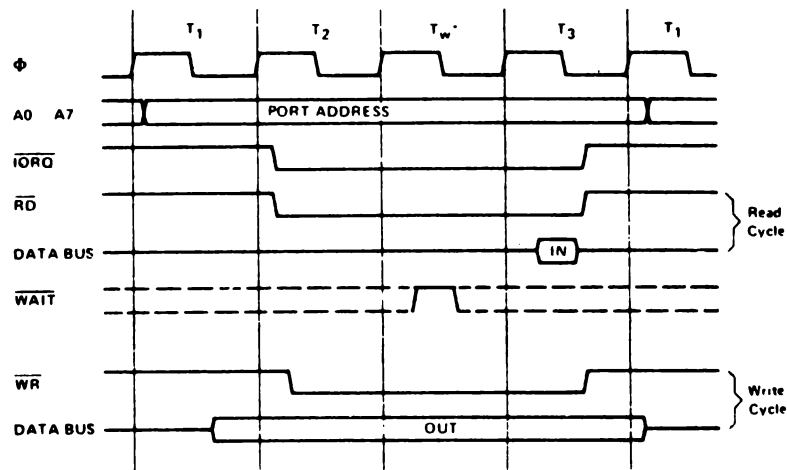
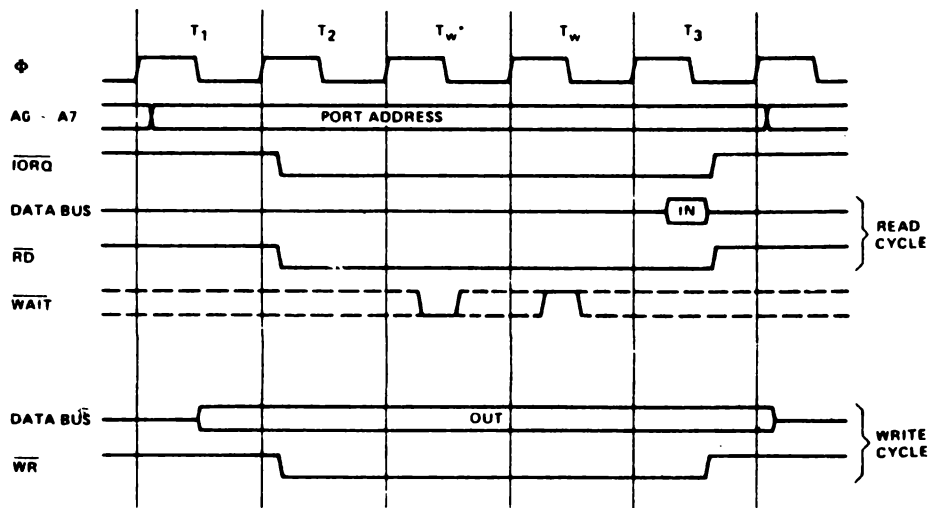
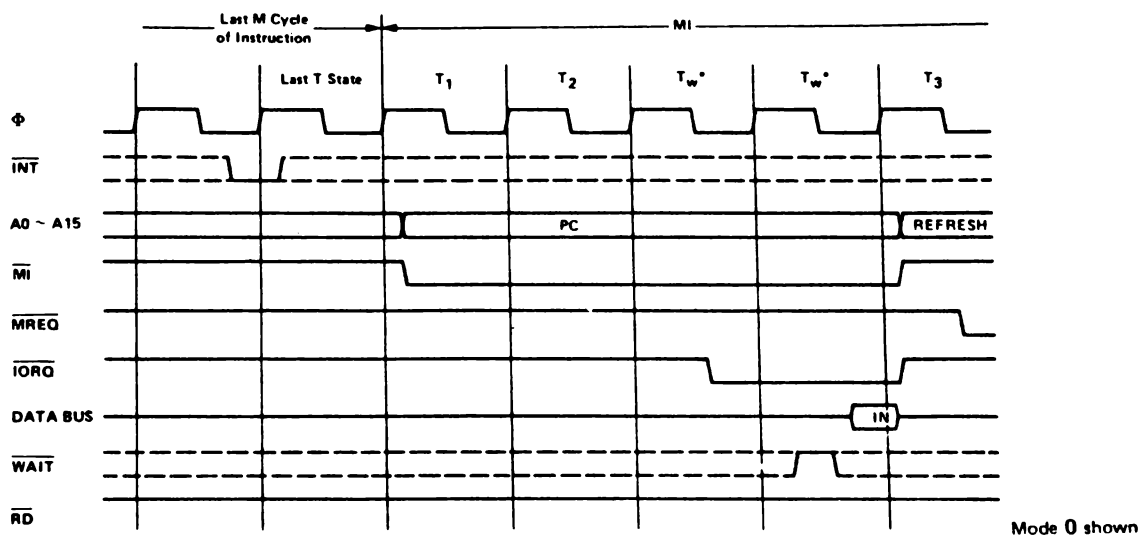
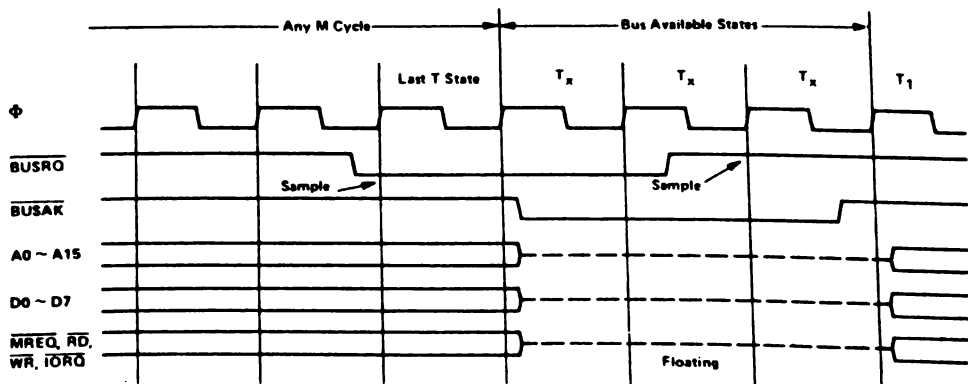
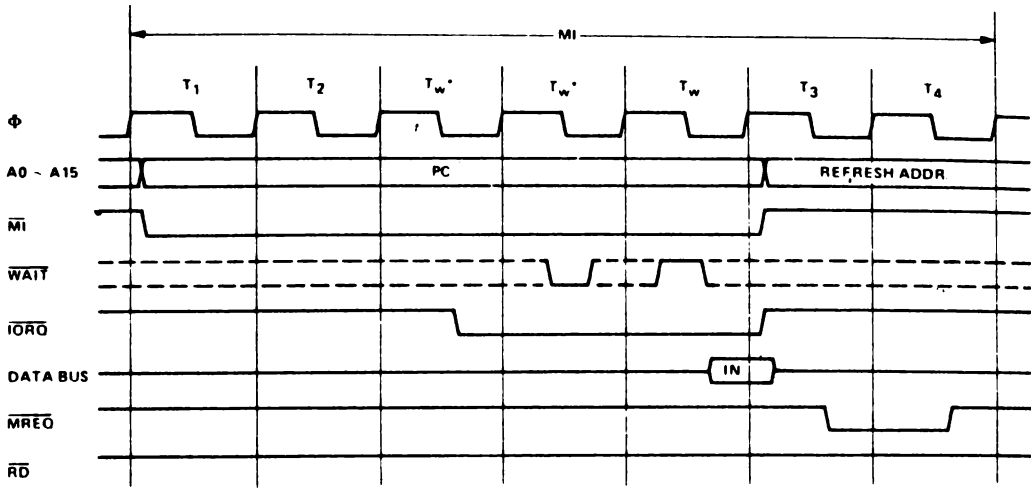
Come si può notare, le uscite $SL_{0\text{-}3}$ sono inviate ad un decodificatore 4/16 il quale attiva, una alla volta, una delle sue 16 uscite, ciclicamente; queste ultime comandano un gruppo di 16 transistor PNP con il compito di fornire la corretta tensione di alimentazione agli anodi di 16 distinti visualizzatori, che nella figura sono del tipo ad anodo comune.
Le 8 uscite $A_{0\text{-}3}$, $B_{0\text{-}3}$, comandano le basi di 8 transistor NPN che hanno il compito di pilotare i catodi (a,b,...,dp) dei 16 visualizzatori, catodi che sono collegati fra loro in parallelo, nel senso che tutti i catodi dei segmenti a sono collegati tra loro, tutti quelli di segmenti b anche e così via. In questo modo in ogni istante, uno solo di visualizzatori è alimentato al suo anodo e quindi saranno luminosi solo quelli, fra i suoi diodi, cui corrisponde una uscita (Ai, Bi) a livello logico alto, in quanto solo in questo modo il corrispettivo transistor NPN è conduttore.
Nel caso siano usati visualizzatori a LED a catodo comune, si può utilizzare uno schema simile a quello di fig. 10.29 adattandolo in modo opportuno.
L'intervallo di tempo in cui un visualizzatore è attivo dipende ovviamente dalla velocità con cui si aggiornano i segnali $S_{0\text{-}3}$ e quindi da quella con cui si effettua la scansione della tastiera; un valore tipico è di 480 μs cui corrisponde un tempo di scansione per la tastiera, e quindi di "rinfresco" dei 16 display, di circa 10 ms.
Occorre notare che lo schema di fig. 10.29 può essere ulteriormente semplificato nel caso che si abbia necessità di solo 4 display e che la scansione della tastiera sia effettuata in modo decodificato; in tal caso infatti alle uscite $SL_{0\text{-}3}$ sono presenti successivamente i valori 0111, 1011, 1101, 1110 e ciò permette di eliminare il decodificatore e di collegare direttamente alle uscite $SL_{0\text{-}3}$ gli ingressi dei transistor PNP. Nel caso si abbia necessità di utilizzare solo 8 visualizzatori, il decodificatore indicato in fig. 10.29 può essere del tipo 3/8 nel quale caso si deve programmare opportunamente l'8279 per indicargli che i codici di scansione del display sono solo 8.
Si noti che nella RAM di display devono essere inviati dalla CPU dei codici opportuni per accendere nel modo voluto i segmenti di un display: opportuni nel senso che se i collegamenti sono quelli indicati in figura, per accendere il segmento a occorre sia posto a 1 il bit meno significativo della locazione in RAM, per il segmento f il bit 5 e così via. E' necessario perciò che la CPU utilizzi, via software, una opportuna tabella, necessaria per convertire le rappresentazioni interne del carattere da visualizzare in quella a 7 segmenti imposta dal circuito utilizzato.
Nel caso si volesse semplificare il software, eliminando la conversione, è possibile modificare lo schema di fig. 10.29, inserendo tra gli ingressi dei transistor NPN e le uscite A_0-A_3 dei convertitori di codice da esadecimale a 7 segmenti. Con questa modifica, se si inseriscono due convertitori,

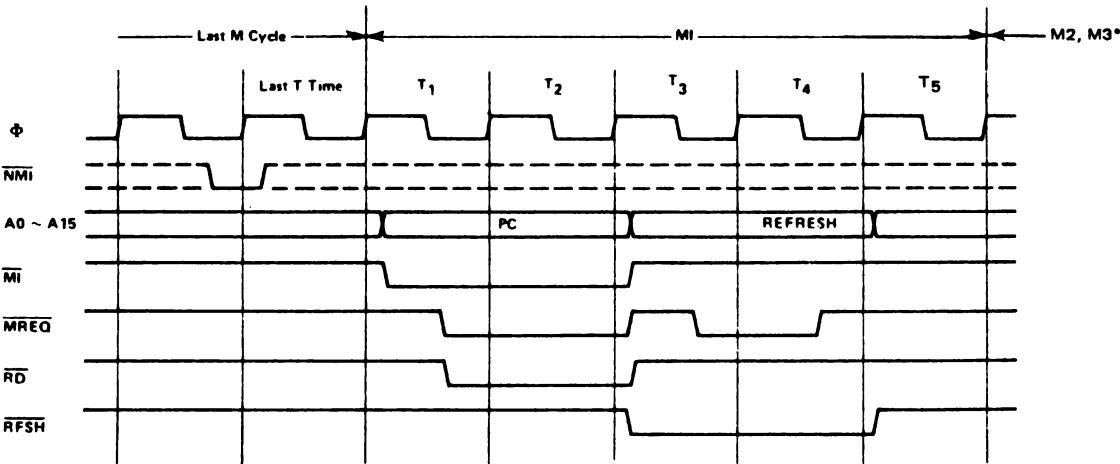
uno collegato alle uscite Ai ed uno alle Bi è anche possibile pilotare addirittura 32 visualizzatori; in tal caso però si deve tener presente che in ogni locazione della RAM di display sono contenuti i caratteri corrispondenti a due visualizzatori e quindi che anche la disposizione fisica di tali visualizzatori deve essere opportunamente organizzata.
E' evidente da quanto detto che l'8279 può pilotare non solo visualizzatori a 7 segmenti: se si ha la necessità di adoperare degli indicatori luminosi per segnalazioni spento-acceso basterà utilizzare, con le stesse modalità viste, dei semplici led singoli.

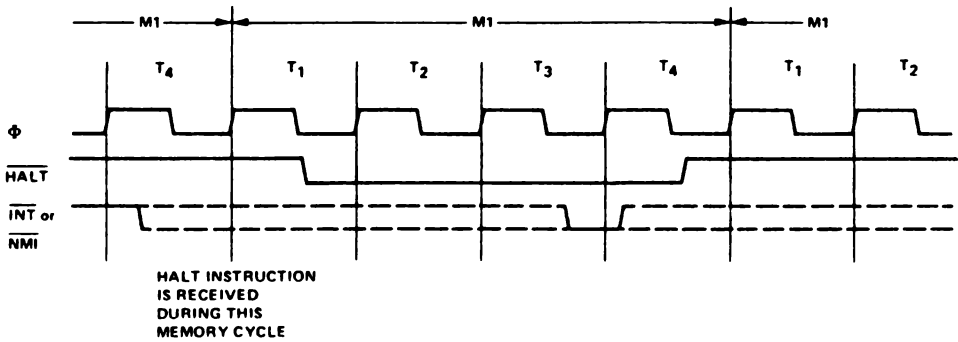
Appendice

CARATTERISTICHE STATICHE E DINAMICHE DEL MICROPROCESSORE Z80

Tavole tratte da:
MOSTEK 1981 - Z80 Microcomputer Data Book

INSTRUCTION OP CODE FETCH

INSTRUCTION OP CODE FETCH WITH WAIT STATES

MEMORY READ OR WRITE CYCLES

MEMORY READ OR WRITE CYCLES WITH WAIT STATES

INPUT OR OUTPUT CYCLES

INPUT OR OUTPUT CYCLES WITH WAIT STATES

INTERRUPT REQUEST/ACKNOWLEDGE CYCLE

BUS REQUEST/ACKNOWLEDGE CYCLE

INTERRUPT REQUEST/ACKNOWLEDGE WITH WAIT STATES

Mode 0 shown

NON MASKABLE INTERRUPT REQUEST OPERATION

*M2 and M3 are stack write operations

HALT EXIT

11.0 ELECTRICAL SPECIFICATIONS
ABSOLUTE MAXIMUM RATINGS*

Temperature Under Bias . . . . .Specified Operating Range

Storage Temperature. . . . .–65°C to +150°C

Voltage on Any Pin with Respect to Ground . . . . –0.3V to +7V

Power Dissipation 1.5W

D.C. CHARACTERISTICS
T_A = 0°C to 70°C, V_{CC} = 5V ± 5% unless otherwise specified

SYMBOL	PARAMETER	MIN.	TYP.	MAX.	UNIT	TEST CONDITION
V_{ILC}	Clock Input Low Voltage	–0.3		0.8	V	
V_{IHC}	Clock Input High Voltage	Vcc-.6		Vcc+.3	V	
V_{IL}	Input Low Voltage	–0.3		0.8	V	
V_{IH}	Input High Voltage	2.0		V_{CC}	V	
V_{OL}	Output Low Voltage			0.4	V	I_{OL} = 1.8mA
V_{OH}	Output High Voltage	2.4			V	I_{OH} = –250 μA
I_{CC}	Power Supply Current			150*	mA	
I_{LI}	Input Leakage Current			±10	μA	V_{IN} = 0 to V_{CC}
I_{LOH}	Tri-State Output Leakage Current in Float			10	μA	V_{OUT} = 2.4 to V_{CC}
I_{LOL}	Tri-State Output Leakage Current in Float			–10	μA	V_{OUT} = 0.4V
I_{LD}	Data Bus Leakage Current in Input Mode			±10	μA	$0 \leq V_{IN} \leq V_{CC}$

*200mA for -4, -10 or -20 devices

NOTE: All outputs are rated at one standard TTL load.

CAPACITANCE
T_A = 25°C, f = 1MHz unmeasured pins returned to ground

SYMBOL	PARAMETER	MAX.	UNIT
CΦ	Clock Capacitance	35	pF
C_{IN}	Input Capacitance	5	pF
C_{OUT}	Output Capacitance	10	pF

*Comment Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other condition above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

A C CHARACTERISTICS

$T_A = 0°C$ to $70°C$, $V_{CC} = +5V \pm 5\%$, Unless Otherwise Noted

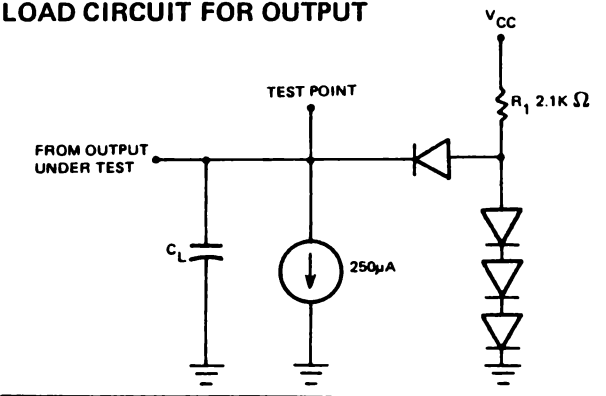
SIGNAL	SYMBOL	PARAMETER	MIN.	MAX.	UNIT	TEST CONDITION
Φ	t_c	Clock Period	.4	[12]	μsec	
	$t_{w(\Phi H)}$	Clock Pulse Width, Clock High	180	(D)	nsec	
	$t_{w(\Phi L)}$	Clock Pulse Width, Clock Low	180	2000	nsec	
	$t_{r,f}$	Clock Rise and Fall Time		30	nsec	
A_{0-15}	$t_{D(AD)}$	Address Output Delay		145	nsec	C_L = 50pF
	$t_{F(AD)}$	Delay to Float		110	nsec	
	t_{acm}	Address Stable Prior to $\overline{MREQ}$ (Memory Cycle)	[1]		nsec	
	t_{aci}	Address Stable Prior to $\overline{IORQ}$, $\overline{RD}$ or $\overline{WR}$ (I/O Cycle)	[2]		nsec	
	t_{ca}	Address Stable From $\overline{RD}$, $\overline{WR}$, $\overline{IORQ}$ or $\overline{MREQ}$	[3]		nsec	Except T3-M1
	t_{caf}	Address Stable From $\overline{RD}$ or $\overline{WR}$ During Float	[4]		nsec	
D_{0-7}	$t_{D(D)}$	Data Output Delay		230	nsec	C_L = 50pF
	$t_{F(D)}$	Delay to Float During Write Cycle		90	nsec	
	$t_{S\Phi(D)}$	Data Setup Time to Rising Edge of Clock During M1 Cycle	50		nsec	
	$t_{S\overline{\Phi}(D)}$	Data Setup Time to Falling Edge at Clock During M2 to M5	60		nsec	
	t_{dcm}	Data Stable Prior to $\overline{WR}$ (Memory Cycle)	[5]		nsec	
	t_{dci}	Data Stable Prior to $\overline{WR}$ (I/O Cycle)	[6]		nsec	
	t_{cdf}	Data Stable From $\overline{WR}$	[7]		nsec	
	t_H	Input Hold Time	0		nsec	
$\overline{MREQ}$	$t_{DL\overline{\Phi}(MR)}$	$\overline{MREQ}$ Delay From Falling Edge of Clock, $\overline{MREQ}$ Low		100	nsec	C_L = 50 pF
	$t_{DH\Phi(MR)}$	$\overline{MREQ}$ Delay From Rising Edge of Clock, $\overline{MREQ}$ High		100	nsec	
	$t_{DH\overline{\Phi}(MR)}$	$\overline{MREQ}$ Delay From Falling Edge of Clock, $\overline{MREQ}$ High		100	nsec	
	$t_{w(\overline{MRL})}$	Pulse Width, $\overline{MREQ}$ Low	[8]		nsec	
	$t_{w(\overline{MRH})}$	Pulse Width, $\overline{MREQ}$ High	[9]		nsec	
$\overline{IORQ}$	$t_{DL\Phi(IR)}$	$\overline{IORQ}$ Delay From Rising Edge of Clock, $\overline{IORQ}$ Low		90	nsec	C_L = 50 pF
	$t_{DL\overline{\Phi}(IR)}$	$\overline{IORQ}$ Delay From Falling Edge of Clock, $\overline{IORQ}$ Low		110	nsec	
	$t_{DH\Phi(IR)}$	$\overline{IORQ}$ Delay From Rising Edge of Clock, $\overline{IORQ}$ High		100	nsec	
	$t_{DH\overline{\Phi}(IR)}$	$\overline{IORQ}$ Delay From Falling Edge of Clock, $\overline{IORQ}$ High		110	nsec	
$\overline{RD}$	$t_{DL\Phi(RD)}$	$\overline{RD}$ Delay From Rising Edge of Clock, $\overline{RD}$ Low		100	nsec	C_L = 50pF
	$t_{DL\overline{\Phi}(RD)}$	$\overline{RD}$ Delay From Falling Edge of Clock, $\overline{RD}$ Low		130	nsec	
	$t_{DH\Phi(RD)}$	$\overline{RD}$ Delay From Rising Edge of Clock, $\overline{RD}$ High		100	nsec	
	$t_{DH\overline{\Phi}(BD)}$	$\overline{RD}$ Delay From Falling Edge of Clock, $\overline{RD}$ High		110	nsec	
$\overline{WR}$	$t_{DL\Phi(WR)}$	$\overline{WR}$ Delay From Rising Edge of Clock, $\overline{WR}$ Low		80	nsec	C_L = 50pF
	$t_{DL\overline{\Phi}(WR)}$	$\overline{WR}$ Delay From Falling Edge of Clock $\overline{WR}$ Low		90	nsec	
	$t_{DH\Phi(WR)}$	$\overline{WR}$ Delay From Falling Edge of Clock, $\overline{WR}$ High		100	nsec	
	$t_{w(\overline{WRL})}$	Pulse Width, $\overline{WR}$ Low	[10]		nsec	

NOTES:

A Data should be enabled onto the CPU data bus when RD is active. During interrupt acknowledge data should be enabled when $\overline{M1}$ and $\overline{IORQ}$ are both active.

B The $\overline{RESET}$ signal must be active for a minimum of 3 clock cycles.

cont'd on page 81

SIGNAL	SYMBOL	PARAMETER	MIN.	MAX.	UNIT	TEST CONDITIONS
$\overline{M1}$	$t_{DL(M1)}$	$\overline{M1}$ Delay From Rising Edge of Clock $\overline{M1}$ Low		130	nsec	C_L = 50pF
	$t_{DH(M1)}$	$\overline{M1}$ Delay From Rising Edge of Clock $\overline{M1}$ High		130	nsec	
$\overline{RFSH}$	$t_{DL(RF)}$	$\overline{RFSH}$ Delay From Rising Edge of Clock, $\overline{RFSH}$ Low		180	nsec	C_L = 30pF
	$t_{DH(RF)}$	$\overline{RFSH}$ Delay From Rising Edge of Clock, $\overline{RFSH}$ High		150	nsec	
$\overline{WAIT}$	$t_{S(WT)}$	$\overline{WAIT}$ Setup Time to Falling Edge of Clock	70		nsec	
$\overline{HALT}$	$t_{D(HT)}$	$\overline{HALT}$ Delay Time From Falling Edge of Clock		300	nsec	C_L = 50pF
$\overline{INT}$	$t_{s(IT)}$	$\overline{INT}$ Setup Time to Rising Edge of Clock	80		nsec	
$\overline{NMI}$	$t_w(\overline{NML})$	Pulse Width, $\overline{NMI}$ Low	80		nsec	
$\overline{BUSRQ}$	$t_{s(BQ)}$	$\overline{BUSRQ}$ Setup Time to Rising Edge of Clock	80		nsec	
$\overline{BUSAK}$	$t_{DL(BA)}$	$\overline{BUSAK}$ Delay From Rising Edge of Clock, $\overline{BUSAK}$ Low		120	nsec	C_L = 50 pF
	$t_{DH(BA)}$	$\overline{BUSAK}$ Delay From Falling Edge of Clock, $\overline{BUSAK}$ High		110	nsec	
$\overline{RESET}$	$t_{s(RS)}$	$\overline{RESET}$ Setup Time to Rising Edge of Clock	90		nsec	
	$t_{F(C)}$	Delay to/from Float ($\overline{MREQ}$, $\overline{IORQ}$, $\overline{RD}$ and $\overline{WR}$)		100	nsec	
	t_{mr}	$\overline{M1}$ Stable Prior to IORQ (Interrupt Ack.)	[11]		nsec	

[1] $t_{acm} = t_w(\Phi H) + t_f - 75$

[2] $t_{aci} = t_c - 80$

[3] $t_{ca} = t_w(\Phi L) + t_r - 40$

[4] $t_{caf} = t_w(\Phi L) + t_r - 60$

[5] $t_{dcm} = t_c - 210$

[6] $t_{dci} = t_w(\Phi L) + t_r - 210$

[7] $t_{cdf} = t_w(\Phi L) + t_r - 80$

[8] $t_w(\overline{MRL}) = t_c - 40$

[9] $t_w(\overline{MRH}) = t_w(\Phi H) + t_f - 30$

[10] $t_w(\overline{WR}) = t_c - 40$

[11] $t_{mr} = 2t_c + t_w(\Phi H) + t_f - 80$

[12] $t_c = t_w(\Phi H) + t_w(\Phi L) + t_r + t_f$

LOAD CIRCUIT FOR OUTPUT

NOTES (Cont'd.)

C. Output Delay vs. Load Capacitance
T_A = 70°C V_{CC} = 5V±5%
Add 10 nsec delay for each 50pF increase in load up to a maximum of 200pF for the data bus and 100pF for address and control lines.

D. Although static by design, testing guarantees $t_w(\Phi H)$ of 200 µsec maximum.

A. C. CHARACTERISTICS T_A = 0°C to 70°C, Vcc = +5V ±5%, Unless Otherwise Noted

SIGNAL	SYMBOL	PARAMETER	MIN.	MAX.	UNIT	TEST CONDITIONS
Φ	t_c	Clock Period	.25	[12]	μsec	
	$t_{w(\Phi H)}$	Clock Pulse Width, Clock High	110	(D)	nsec	
	$t_{w(\Phi L)}$	Clock Pulse Width, Clock Low	110	2000	nsec	
	$t_{r,f}$	Clock Rise and Fall Time		30	nsec	
A_{0-15}	$t_{D(AD)}$	Address Output Delay		110	nsec	
	$t_{F(AD)}$	Delay to Float		90	nsec	
	t_{acm}	Address Stable Prior to $\overline{MREQ}$ (Memory Cycle)	[1]		nsec	C_L = 50pF
	t_{aci}	Address Stable Prior to $\overline{IORQ}$, $\overline{RD}$ or $\overline{WR}$ (I/O Cycle)	[2]		nsec	
	t_{ca}	Address Stable From $\overline{RD}$, $\overline{WR}$, $\overline{IORQ}$ or $\overline{MREQ}$	[3]		nsec	Except T3.M1
	t_{caf}	Address Stable From $\overline{RD}$ or $\overline{WR}$ During Float	[4]		nsec	
D_{0-7}	$t_{D(D)}$	Data Output Delay		150	nsec	
	$t_{F(D)}$	Delay to Float During Write Cycle		90	nsec	
	$t_{S\Phi(D)}$	Data Setup Time to Rising Edge of Clock During M1 Cycle	35		nsec	
	$t_{S\overline{\Phi}(D)}$	Data Setup Time to Falling Edge at Clock During M2 to M5	50		nsec	C_L = 50pF
	t_{dcm}	Data Stable Prior to $\overline{WR}$ (Memory Cycle)	[5]		nsec	
	t_{dci}	Data Stable Prior to $\overline{WR}$ (I/O Cycle)	[6]		nsec	
	t_{cdf}	Data Stable From $\overline{WR}$	[7]		nsec	
	t_H	Input Hold Time	0		nsec	
$\overline{MREQ}$	$t_{DL\overline{\Phi}(MR)}$	$\overline{MREQ}$ Delay From Falling Edge of Clock, $\overline{MREQ}$ Low	20	85	nsec	
	$t_{DH\Phi(MR)}$	$\overline{MREQ}$ Delay From Rising Edge of Clock, $\overline{MREQ}$ High		85	nsec	
	$t_{DH\overline{\Phi}(MR)}$	$\overline{MREQ}$ Delay From Falling Edge of Clock, $\overline{MREQ}$ High		85	nsec	C_L = 50pF
	$t_{w(\overline{MRL})}$	Pulse Width, $\overline{MREQ}$ Low	[8]		nsec	
	$t_{w(\overline{MRH})}$	Pulse Width, $\overline{MREQ}$ High	[9]		nsec	
$\overline{IORQ}$	$t_{DL\Phi(IR)}$	$\overline{IORQ}$ Delay From Rising Edge of Clock, $\overline{IORQ}$ Low		75	nsec	
	$t_{DL\overline{\Phi}(IR)}$	$\overline{IORQ}$ Delay From Falling Edge of Clock, $\overline{IORQ}$ Low		85	nsec	C_L = 50pF
	$t_{DH\Phi(IR)}$	$\overline{IORQ}$ Delay From Rising Edge of Clock, $\overline{IORQ}$ High		85	nsec	
	$t_{DH\overline{\Phi}(IR)}$	$\overline{IORQ}$ Delay From Falling Edge of Clock, $\overline{IORQ}$ High		85	nsec	
$\overline{RD}$	$t_{DL\Phi(RD)}$	$\overline{RD}$ Delay From Rising Edge of Clock, $\overline{RD}$ Low		85	nsec	
	$t_{DL\overline{\Phi}(RD)}$	$\overline{RD}$ Delay From Falling Edge of Clock, $\overline{RD}$ Low		95	nsec	C_L = 50pF
	$t_{DH\Phi(RD)}$	$\overline{RD}$ Delay From Rising Edge of Clock, $\overline{RD}$ High		85	nsec	
	$t_{DH\overline{\Phi}(RD)}$	$\overline{RD}$ Delay From Falling Edge of Clock, $\overline{RD}$ High		85	nsec	
$\overline{WR}$	$t_{DL\Phi(WR)}$	$\overline{WR}$ Delay From Rising Edge of Clock, $\overline{WR}$ Low		65	nsec	
	$t_{DL\overline{\Phi}(WR)}$	$\overline{WR}$ Delay From Falling Edge of Clock, $\overline{WR}$ Low		80	nsec	C_L = 50pF
	$t_{DH\Phi(WR)}$	$\overline{WR}$ Delay From Falling Edge of Clock, $\overline{WR}$ High		80	nsec	
	$t_{w(\overline{WRL})}$	Pulse Width, $\overline{WR}$ Low	[10]		nsec	

NOTES:

A Data should be enabled onto the CPU data bus when $\overline{RD}$ is active. During interrupt acknowledge data should be enabled when M1 and IORQ are both active.

B The $\overline{RESET}$ signal must be active for a minimum of 3 clock cycles.

(Cont'd. on page 83)

SIGNAL	SYMBOL	PARAMETER	MIN.	MAX.	UNIT	TEST CONDITION
$\overline{M1}$	$t_{DL(M1)}$	$\overline{M1}$ Delay From Rising Edge of Clock $\overline{M1}$ Low		100	nsec	$C_L = 50pF$
	$t_{DH(M1)}$	$\overline{M1}$ Delay From Rising Edge of Clock, $\overline{M1}$ High		100	nsec	
$\overline{RFSH}$	$t_{DL(RF)}$	$\overline{RFSH}$ Delay From Rising Edge of Clock, $\overline{RFSH}$ Low		130	nsec	$C_L = 50pF$
	$t_{DH(RF)}$	$\overline{RFSH}$ Delay From Rising Edge of Clock $\overline{RFSH}$ High		120	nsec	
$\overline{WAIT}$	$t_{S(WT)}$	$\overline{WAIT}$ Setup Time to Falling Edge of Clock	70		nsec	
$\overline{HALT}$	$t_{D(HT)}$	$\overline{HALT}$ Delay Time From Falling Edge of Clock		300	nsec	$C_L = 50pF$
$\overline{INT}$	$t_{s(IT)}$	$\overline{INT}$ Setup Time to Rising Edge of Clock	80		nsec	
$\overline{NMI}$	$t_{w(\overline{NML})}$	Pulse Width, $\overline{NMI}$ Low	80		nsec	
$\overline{BUSRQ}$	$t_{s(BQ)}$	$\overline{BUSRQ}$ Setup Time to Rising Edge of Clock	50		nsec	
$\overline{BUSAK}$	$t_{DL(BA)}$	$\overline{BUSAK}$ Delay From Rising Edge of Clock, $\overline{BUSAK}$ Low		100	nsec	$C_L = 50pF$
	$t_{DH(BA)}$	$\overline{BUSAK}$ Delay From Falling Edge of Clock, $\overline{BUSAK}$ High		100	nsec	
$\overline{RESET}$	$t_{s(RS)}$	$\overline{RESET}$ Setup Time to Rising Edge of Clock	60		nsec	
	$t_{F(C)}$	Delay to/From Float ($\overline{MREQ}$, $\overline{IORQ}$, $\overline{RD}$ and $\overline{WR}$)		80	nsec	
	t_{mr}	$\overline{M1}$ Stable Prior to $\overline{IORQ}$ (Interrupt Ack.)	[11]		nsec	

LOAD CIRCUIT FOR OUTPUT

[1] $t_{acm} = t_w(\Phi H) + t_f - 65$

[2] $t_{aci} = t_c - 70$

[3] $t_{ca} = t_w(\Phi L) + t_r - 50$

[4] $t_{caf} = t_w(\Phi L) + t_r - 45$

[5] $t_{dcm} = t_c - 170$

[6] $t_{dci} = t_w(\Phi L) + t_r - 170$

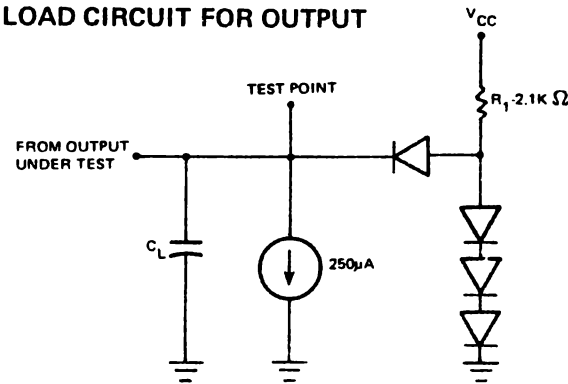
[7] $t_{cdf} = t_w(\Phi L) + t_r - 70$

[8] $t_w(\overline{MRL}) = t_c - 30$

[9] $t_w(\overline{MRH}) = t_w(\Phi H) + t_f - 20$

[10] $t_w(\overline{WR}) = t_c - 30$

[11] $t_{mr} = 2t_c + t_w(\Phi H) + t_f - 65$

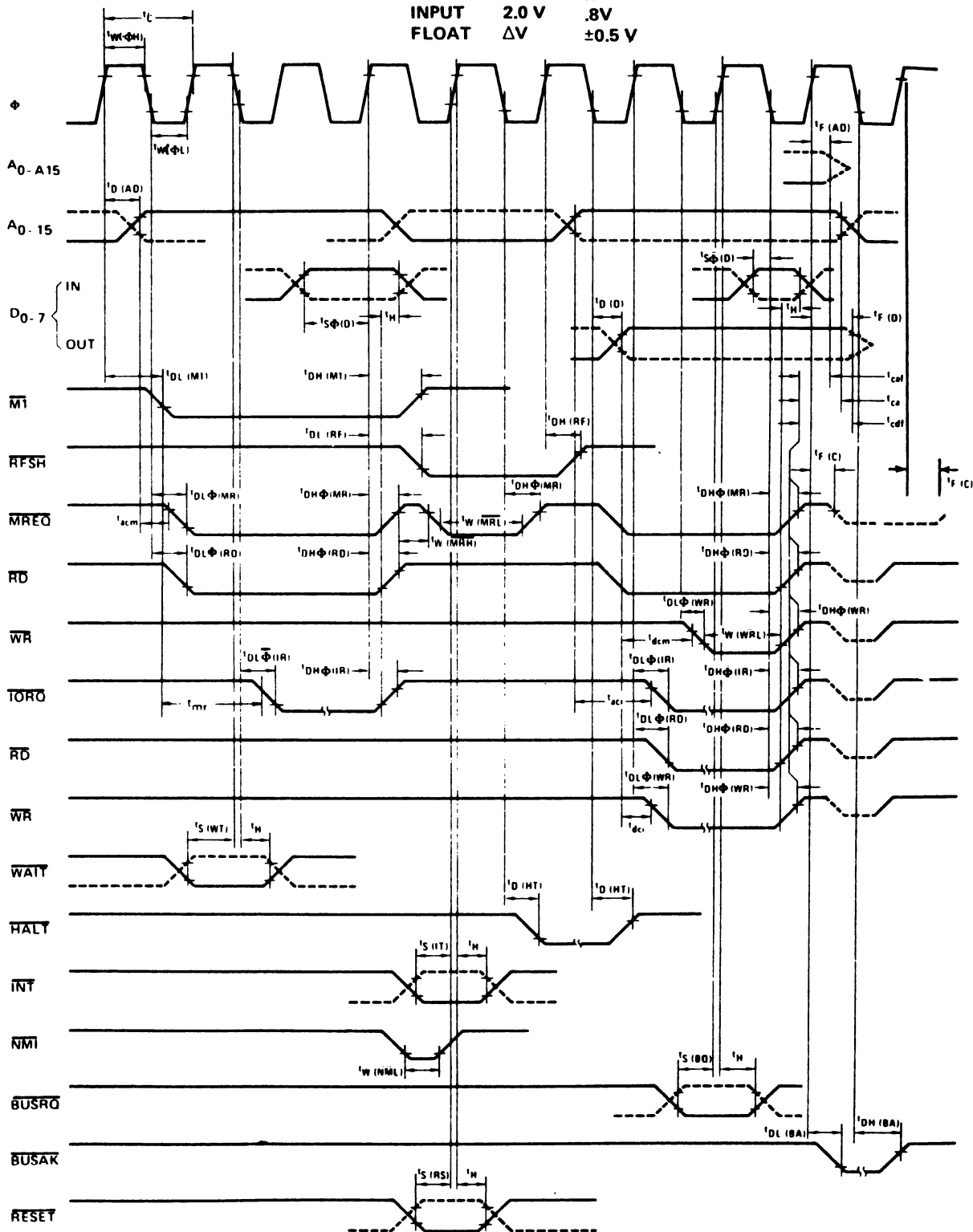
[12] $t_c = t_w(\Phi H) + t_w(\Phi L) + t_r + t_f$

NOTES (Cont'd.)

C. Output Delay vs. Load Capacitance
$T_A = 70^\circ C$ $V_{CC} = 5V \pm 5\%$
Add 10 nsec delay for each 50pF increase in load up to a maximum of 200pF for the data bus and 100pF for address and control lines

D. Although static by design, testing guarantees $t_w(\Phi H)$ of 200 μsec maximum.

A.C. TIMING DIAGRAM

Timing measurements are made at the following voltages, unless otherwise specified:

	"1"	"0"
CLOCK	V_{cc}–.6	.8V
OUTPUT	2.0 V	.8V
INPUT	2.0 V	.8V
FLOAT	ΔV	±0.5 V

Serie
Informatica

Giancarlo Baccolini
Carlo Offelli

MICROELABORATORI
Note di hardware

Il volume rappresenta la logica continuazione di quanto esposto nel precedente, Microelaboratori: fondamenti. *Desiderio degli Autori è fornire le indicazioni di carattere generale per la realizzazione di un microelaboratore. Sono perciò messe in risalto le modalità di scambio d'informazione fra i vari dispositivi presenti in un elaboratore, per organizzare la struttura hardware richiesta in una qualsiasi applicazione.*
Caratteristiche fondamentali del volume sono la completezza e la leggibilità.
E' indubbio, infatti, che oggi sono reperibili numerosi testi aventi come argomento i microprocessori e le problematiche associate: quest'opera non si ferma però alla semplice elencazione dei problemi, ma vuol dare al lettore uno strumento in grado di fornire risposte "reali" agli altrettanto reali problemi tecnici.
Un grande lavoro di chiarezza e concisione, con il supporto di una profonda conoscenza della materia, permette di addentrarsi in argomentazioni tecniche sofisticate mantenendosi a un livello di completa leggibilità.

ISBN 88 - 7005 - 558 - 2

lire ventimila (i.i.)